UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

Deutsches Zentrum
DLR für Luft- und Raumfahrt
German Aerospace Center

FINAL DEGREE PROJECT

TFG TITLE: Validation using performance reference data of an Airbus A320 performance model for flight simulation

DEGREE: Bachelor's degree in Air Navigation Engineering

AUTHORS: Raúl Sáez García

ADVISOR: Helge Lenz

SUPERVISOR: Xavier Prats i Menéndez

DATE: October 23, 2015

Títol: Validació del model de *performance* d'un Airbus A320 amb dades de referència de *performance* per simulacions de vol

Autors: Raúl Sáez García

Director: Helge Lenz

Supervisor: Xavier Prats i Menéndez

Data: 23 d'octubre de 2015

Resum

L'objectiu d'aquest projecte és millorar el model de *performance* del QPAC (QualityPark AviationCenter) A320 al simulador de vol X-Plane. Per aconseguir-ho s'han fet servir taules generades amb l'Airbus PEP (*Performance Engineering Program tool*). El procés que s'ha seguit consisteix en utilitzar les taules del PEP com a taules de referència pel simulador, de manera que a través d'un *plugin* és possible moure aquestes dades del PEP a l'X-Plane. És de gran interès tenir un model de *performance* correcte al simulador, ja que això ens conduirà a uns millors resultats a l'hora de fer simulacions per provar nous conceptes aeronàutics. En aquest projecte es mostrarà com a exemple com la millora del model permet obtenir millors resultats en una simulació de CDO (*Continuous Descent Operations*) amb l'X-Plane. El model de *performance* d'X-Plane està basat en el que es coneix com *Blade element theory*, la qual cosa és una de las grans diferències amb els altres simuladors de vol presents avui en dia. Tot i que els resultats són prou bons a la majoria dels aspectes de la simulació, hi ha grans imprecisions quan es tracta de modelar efectes transònics i supersònics. A més, els valors de *thrust* tampoc són gaire bons. Les correccions de *performance* que s'han aplicat són de *drag* i d'*idle thrust*. Les taules del PEP s'han mogut a arxius anomenats *headers* a C++ de manera que el *plugin* d'X-Plane les pugui llegir fàcilment. Pel cas del *drag*, es llegeixen coeficients de *drag* donat un número de Mach i un coeficient de *lift* com a valors d'entrada mentre que pel cas de l'*idle thrust*, es llegeixen valors d'*idle thrust* donat un número de Mach i una altitud com a valors d'entrada. Després d'aplicar les correccions, s'ha observat una gran millora tant en els valors de *drag* com en els d'*idle thrust*, amb errors baixos al comparar-ho amb les dades del PEP. Tot i així, encara hi ha algunes imprecisions al model, de manera que es necessitarà més treball en un futur.

Title : Validation using performance reference data of an Airbus A320 performance model for flight simulation

Authors: Raúl Sáez García

Advisor: Helge Lenz

Supervisor: Xavier Prats i Menéndez

Date: October 23, 2015

Overview

The aim of this project is to improve the performance model of the QPAC (QualityPark AviationCenter) A320 in X-Plane flight simulator. In order to achieve it, tables generated with Airbus PEP (Performance Engineering Program tool) have been used. The process followed consists in using PEP tables as look-up tables for the simulator, so that via a plugin it is possible to move this data from PEP to X-Plane. It is of great interest to have a correct performance model in the simulator, as it will lead to more accurate results when testing new aeronautical concepts with flight simulations. In this project it will be shown as an example how the improvement of the model allows to obtain better results in a CDO (Continuous Descent Operations) simulation with X-Plane. The performance model in X-Plane is based in blade element theory, which is one of the big differences with the other flight simulators available nowadays. Although the results are quite good in most of the aspects of the simulation, there are great inaccuracies when it comes to model transonic and supersonic effects. In addition, thrust values are not very good neither. The performance corrections have been applied on drag and idle thrust. PEP tables have been moved to header files in C++ so that an X-Plane plugin can read them easily. In the drag case, drag coefficients are read given Mach number and lift coefficient as input values while in the idle trust case, idle thrust values are read given Mach number and altitude as input values. After applying the corrections, it has been observed a big improvement in both drag and idle thrust values, with low errors when comparing with PEP data. However, there are still some inaccuracies in the model, so some future work will be needed.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

With the evolution of modern high-performance aircraft, experimental costs and increasing flight safety issues, the importance of flight simulators has increased a lot in the last decades. They are not only used for pilot training, but also for other applications such as flight planning, envelope expansion, design and analysis of control laws, handling qualities investigations, and pilot-in-the-loop studies. Furthermore, flight simulators provide a reliable, safe and economical testbed for other investigations, like for instance flight-control research and system assessment, apart from providing also means to verify new aircraft design concepts. Flight simulators mean a reduction in the development time and costs of new aircraft technologies.

In order to obtain accurate results with flight simulations, it is also necessary to have an accurate flight model of the aircraft. For instance, if it is wanted to test some approaches but the drag model is incorrect, the resulting trajectory can be either below or above the real trajectory. Nowadays, there are several flight simulators available in the market, but most of them offer a poor performance model that make them not to be accurate enough in some situations. Therefore, it is needed an improvement in the performance model, which will lead to better results from the simulations. Currently, there are three types of flight simulators: open source, closed, and those which, although they are closed too, they offer the possibility to change several aspects of the simulator by using APIs (Application Programming Interfaces), specific interfaces or similar tools. X-Plane, from Laminar Research, is included in the third group. It allows to read and write data to the simulator via a plugin, which is an executable code written in C++. By using X-Plane, the user can access a great amount of flight parameters, such as aerodynamic coefficients, forces or moments.

X-Plane performance model is based in blade element theory. The main idea of this mathematical process consists in breaking the wing into several small parts so that it is computed the forces acting on them. Then, these forces are integrated along the whole wing to obtain the total force. The same process is repeated for the different parts of the aircraft. Although the performance model works well in several situations there are some inaccuracies when modeling transonic and supersonic effects. Compressible flow effects are not properly considered neither and, as a result of this, the performance data obtained during the simulations is not accurate. Some papers discuss these issues ([5] and [6]), but none of them offer a solution to the problem. Furthermore, there is lot of useful information in forums on the Internet ([7] and [8]), where X-Plane users ask and also offer their help on different X-Plane issues. However, a clear solution to the performance model problem cannot be found in these sources. All the information available only helps to have a better understanding of how X-Plane works, and which are the main problems of it.

The main objective of this project is to improve the performance model of X-Plane, and more specifically the model from the QPAC (QualityPark AviationCenter) A320. This aircraft has been developed by QPAC for X-Plane and it offers a great simulation of the several systems of the aircraft, among other issues. The problem is that the performance model is the one from X-Plane, so the inaccuracies are still there. However, QPAC has developed a plugin that allows to change some simulator parameters that otherwise would remain unchangeable. Therefore, with both X-Plane and QPAC it will be possible to change the performance model of the simulator. Apart from this, it will be used also PEP (Performance Engineering Program tool) from Airbus, which will provide several tables with performance

reference data which will be moved to X-Plane via a new plugin developed for this prupose. It has to be remarked that in this project the parts of the performance model that have been changed are the drag and the idle thrust models.

The project is divided into five chapters. Chapter 1, Fundamentals, is thought to give some theoretical background needed for this project. It is given information about the X-Plane flight model and its limitations, and also explanations of two of the possible methods to obtain performance data that can be later moved to X-Plane, parameter identification from flight test data and PEP tool. In Chapter 2, Problem definition and approach, the basic objective of this project will be defined in detail, and also the different steps followed to achieve it. In Chapter 3, X-Plane plugin, information about X-Plane plugins will be provided, and also specific information about the new plugin developed in this project. In Chapter 4, Tests and results, it will be shown the different tests done with X-Plane to change the model, and the results obtained after doing the correction. Finally, in Chapter 5, Application on Continuous Descent Operations (CDO) trajectories, it will be shown how the improvement of the X-Plane model allows to obtain more accurate results in a simulation of a current aeronautical concept such as CDO.

CHAPTER 1. FUNDAMENTALS

1.1. Aerodynamic coefficients determination

One of the biggest challenges in aerospace engineering and more specifically in flight mechanics is determining an aerodynamic model of a high-performance, highly augmented vehicle from rapid, large-amplitude maneuvers[1]. The problem is that most of the time this model is of unknown structure, highly nonlinear and affected by elastic modes, unsteady aerodynamics and erroneous air data measurements. In flight mechanics, the main objective is to predict and evaluate the performance and dynamical characteristics of a flight vehicle. Representation of motion of a flight vehicle, which generally moves in any direction, involves coupled equations of motion. These equations assume that the forces and moments acting on a vehicle can be synthesized. The validity and accuracy of mathematical models depend on the accuracy on which these external forces and moments can be modeled. These forces and moments can be divided into aerodynamic, inertial, gravitational and propulsive forces.

The aerodynamic modeling provides a means to obtain a relationship between the forces and the moments, and also the possibility to obtain aerodynamic coefficients. In general, it can be said that there are three techniques of determining these coefficients:

- Analytical methods.
- Wind-tunnel methods.
- Flight test methods.

The first two methods are used to obtain basic information about the flight mechanical parameters. However, the analytical methods have a doubtful validity and also have the disadvantage of being used on inadequate theory. Nevertheless, it has to be remarked that computational fluid dynamics have provided in the recent years numerical solutions to complete configurations via Euler and Navier-Stokes flow solvers. Regarding wind-tunnel methods, although providing a great amount of data for different flight vehicle configurations, have certain problems and limitations such as model scaling, Reynold's number, dynamic derivatives, cross coupling and aeroservoelasticity effects. Additionally, these methods can be affected by tunnel unsteadiness or model support vibrations and interferences. It becomes clear that determination of aerodynamic derivatives from flight measurements is one of the best solutions in order to reduce the limitations of the other methods. This section will be focused on giving a general idea of the parameter identification from flight test data, although there is also parameter identification from analytical and wind-tunnel methods.

Parameter identification is a very important step towards the improvement of dynamic models from very different sources. Identifying parameters from flight test data has a great engineering utility as it allows to determine stability and control derivatives, lift and drag characteristics, investigate the structural dynamics of the aircraft and, also, study propulsion effects. Aircraft parameter identification is related to the flight test verification of qualitative (model) and quantitative (coefficients) aerodynamics from a flight mechanics point of view. Hence, aircraft parameter identification can be subdivided into two distinct areas[9]:

- **System identification**: primarily concerned with the mathematical structure of aircraft models.
- **Parameter estimation**: focused on the quantifying of parameters or coefficients for a selected aircraft model.

More precisely defined, system identification is the following[10]: *"process of determining a model structure and related model parameter of a dynamic system with known system excitation and response. It is used to derive a mathematical model and the governing parameters from experimental flight test data."*

In figure 1.1, it can be seen the application spectrum of system identification and also the different existing methods with their complexity and accuracy.

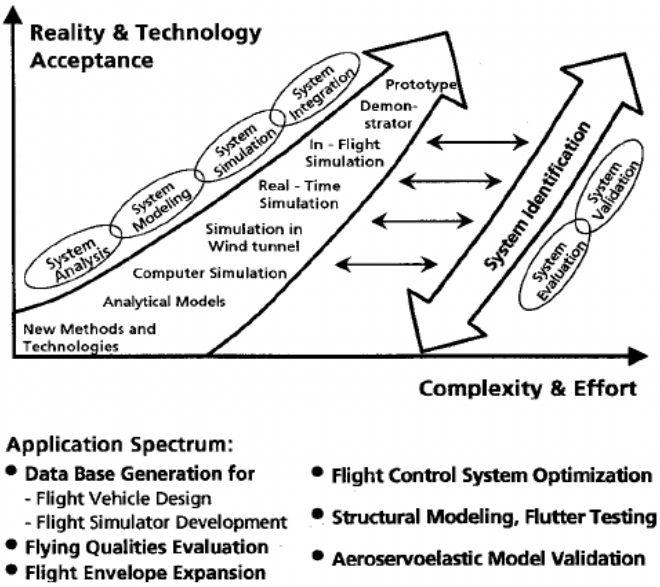


Figure 1.1: Key role of System Identification (reference [1])

System identification is based on an idea called Quad-M process. It is based on the following concepts:

- Maneuver design.
- Measurement accuracy.
- Method definition.
- Model definition.

In figure 1.2, it can be seen the general process flow of the Quad-M process[11].

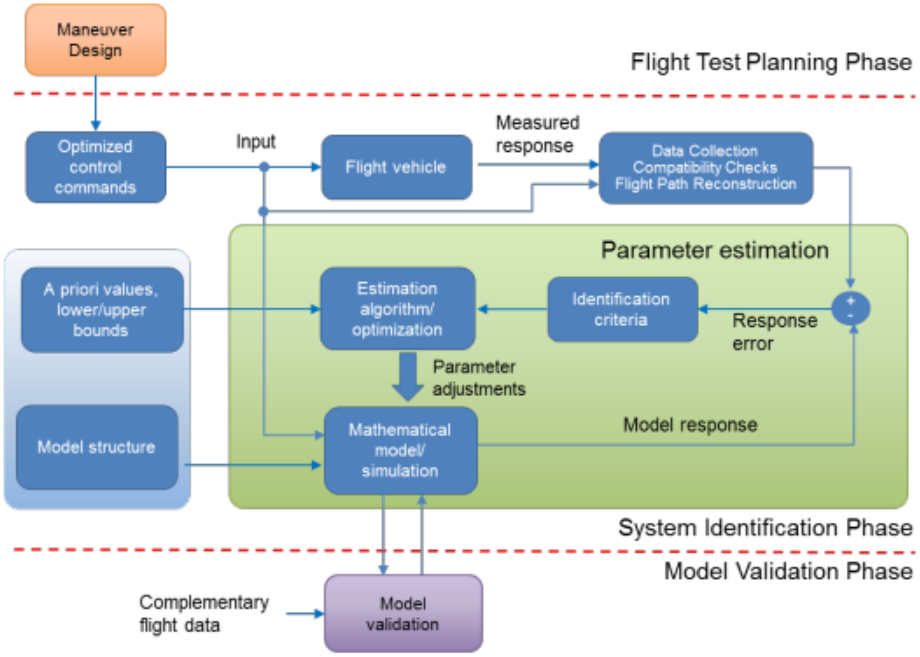


Figure 1.2: Quad-M Process Flow Diagram

The first step in system identification is the maneuver design. It consists in the design of a flight test program where the aircraft is excited in such a way that the measured outputs contain a maximum of information about the aircraft characteristics. Flight tests are used to determine the aerodynamic derivatives, while other issues like the mass properties and the actuator dynamics are determined in laboratory tests. Once the test has been performed, it is necessary to collect all the data and measure the state variables.

With the data gathered it is possible to develop a model. To do it, it is necessary to use a method in order to make adjustments in the model and being able to optimize until a satisfactory result is obtained. It has to be remarked that the model can be implemented with different degrees of fidelity: it either can be a linear state-space model or a high-fidelity six degree of freedom non-linear model. This part is known as parameter estimation. In a final step, the model has to be validated, in order to see whether the specifications are met or not.

In figure 1.3, it can be seen an example of some of the parameters that are recorded during a flight test, more specifically in flight tests done with the ATRA (Advanced Technology Research Aircraft) in the German Aerospace Center in Braunschweig. It is a very important step to decide which will be the parameters that will be recorded; there has to be enough parameters (and the right ones) so that it is possible to obtain an aerodynamic model of the aircraft from them.

```
_3412131100:»   STATIC AIR TEMPERATURE
_3412151100:»   IMPACT PRESSURE
_3412201100:»   BARO CORRECTED ALTITUDE #2
_3412211100:»   INDICATED ANGLE OF ATTACK
_3412341100:»   BARO CORRECTION MILLIBAR #1
_3412361100:»   BARO CORRECTION MILIBAR #2
_3412411100:»   CORRECTED ANGLE OF ATTACK
_3412421100:»   TOTAL PRESSURE
_3412461100:»   CORRECTED AVERAGED STATIC PRESSURE
_3420441100:»   TRUE HEADING
_3420761100:»   GPS ALTITUDE (IR 1)
_3421101100:»   GPS LATITUDE (IR 1)
_3421111100:»   GPS LONGITUDE (IR 1)
_3421321100:»   HYBRID TRUE HEADING (IR 1)
_3421351100:»   HYBRID VERTICAL FIGURE OF MERIT (IR 1)
_3421361100:»   VERTICAL FIGURE OF MERIT (IR 1)
_3421371100:»   HYBRID TRACK ANGLE (IR 1)
_3421651100:»   GPS VERTICAL VELOCITY (IR 1)
_3421751100:»   HYBRID GROUND SPEED (IR 1)
_3422541100:»   HYBRID LATITUDE (IR 1)
_3422551100:»   HYBRID LONGITUDE (IR 1)
_3422561100:»   HYBRID LATITUDE FINE (IR 1)
_3422571100:»   HYBRID LONGITUDE FINE (IR 1)
_3422611100:»   HYBRID ALTITUDE (IR 1)
_3422631100:»   HYBRID FLIGHT PATH ANGLE (IR 1)
_3422641100:»   HYBRID HORIZONTAL FIGURE OF MERIT (IR 1)
_3422661100:»   HYBRID NORTH VELOCITY (IR 1)
_3422671100:»   HYBRID EAST VELOCITY (IR 1)
_3423101100:»   PRESENT POSITION LATITUDE
_3423111100:»   PRESENT POSITION LONGITUDE
_3423121100:»   GROUND SPEED
```

Figure 1.3: Flight test data recorded parameters

There are a lot of available methods that can be used in the parameter identification process. The most remarkable ones are the following:

- Linear least squares.
- Weighted least squares.
- Steady-state oscillator analysis and Fourier analysis(frequency response methods).
- Time-vector technique.
- Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). [12]
- Modified maximum likelihood estimation.

In Dryden NASA they developed the last method mentioned, called Maximum likelihood estimation method[2]. It will be taken as an example to have a better understanding of how parameter identification works. In figure 1.4, it is shown its basic diagram.

On a first step, the measured vehicle response is compared with the simulated response, and the difference between these responses is called the response error. A cost function includes this response error. A modified Newton-Raphson minimization algorithm is used to find the coefficient values (the stability and control derivatives) that minimize the cost function. In each iteration the algorithm provides a new estimate of the coefficients on the basis of the response error. The coefficients of the mathematical model are updated with these estimates, which provides a new estimated response and, therefore, a new response error. When a convergence is reached the updates on the mathematical model are finished. The estimates obtained from this procedure are the maximum-likelihood estimates.

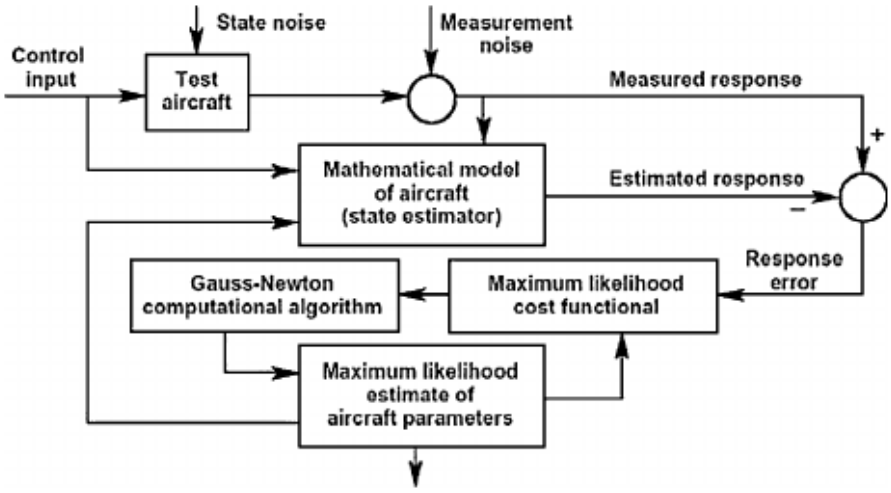


Figure 1.4: Maximum likelihood method diagram (reference [2])

More information about all the current methods and of parameter identification in general can be found in reference [13], where it has been gathered some of the most interesting papers and publications regarding this topic. It is important to highlight too the work made by Ravindra Jategaonkar, who has contribute a lot to the field of system identification and whose documents and papers have helped a lot to make this introduction to parameter identification.

Although parameter identification is a very important step to obtain an accurate flight model, it is also a very time consuming technique. Additionally, it requires a lot of experience in aerospace engineering and a good understanding of the flight characteristics of the aircraft being studied. In chapter 2, it will be discussed more in detail different issues of this technique and also if it is worth or not using it to solve the problem being faced in this project.

1.2. X-Plane

X-Plane is a flight simulator developed by Laminar Research. Together with X-Plane, there are two other software called Plane Maker and Airfoil Maker. The first one, in a few words, is used to design and customize aircraft and all its parts (wings, fuselage,...). It can be used either to create a new aircraft or to change some aspects of existing ones. Airfoil Maker is used exclusively to create and customize airfoils, that can be used after in Plane Maker to create the wings. All the information regarding this software can be found in the corresponding manuals [14] [15].

This section is thought to give a general understanding of how X-Plane works. It is important to have clear some concepts about the X-Plane flight model and its limitations. Furthermore, information of the aircraft tested in the simulator will be given.

1.2.1. Flight Model

X-Plane does not work as the major part of the current flight simulators. While most of them are based on stability derivatives and look-up tables, X-Plane is based on blade element theory [3] [16], which is an engineering process that was initially designed in 1878 by William Froude to determine the behavior of propellers and predicting their performance. Some years after, Stefan Drzewiecki did a more rigorous analysis of this theory, during his work developed between 1892 and 1920 [17] [18]. The basic idea of this theory consists in breaking the propeller blade into a discrete number of sections along the length, like shown in figure 1.5.

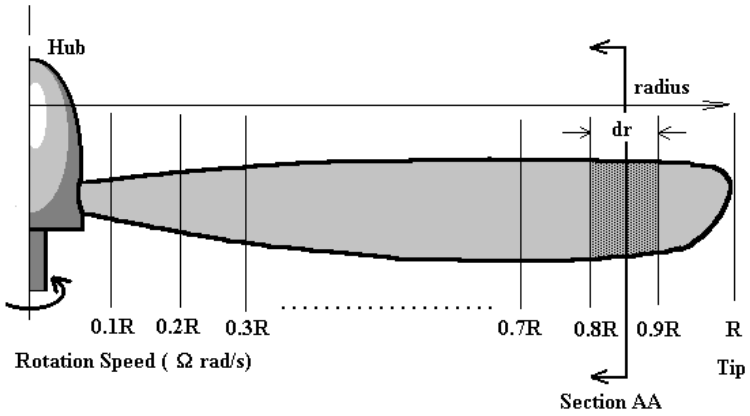


Figure 1.5: Propeller blade divided into small pieces (reference [3])

Then, a force balance is applied in each section, involving 2D lift and drag with the thrust and torque produced too. At the same time, it is applied a balance of axial and angular momentum. All this produces a set of non-linear equations that have to be solved by iteration for each blade section. Once the results are obtained, it is possible to sum them in order to find the overall performance of the propeller. In figure 1.6 it can be seen the analysis of the flow components in the section AA of figure 1.5.

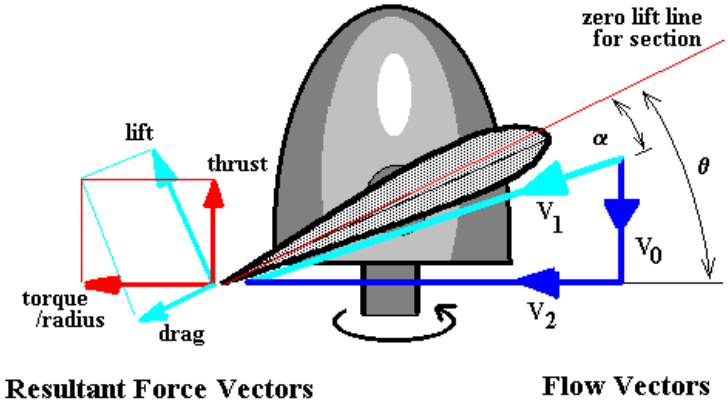


Figure 1.6: Propeller section flow analysis (reference [3])

In X-Plane this theory is applied in the whole aircraft [19], with a greater focus in the wings and the propellers. The simulator analyses the geometric shape of the aircraft and, by using the blade element theory, it splits it into different parts, computing the forces acting in each of these parts many times per second. So, for instance, if it is wanted to know which is the lift generated by a wing, it is only needed to split it in different sections and compute the lift for each one, like the process followed in the propeller blade case. In order to know into how many parts the wing is divided, it can be checked in Plane Maker. In figure 1.7 it can be seen that there are different wing sections and for example in Wing 2 there are 5 parts (highlighted in red). The user can decide the number of parts per wing, to a maximum of 10; according to Laminar Research this quantity is more than enough as a bigger one will lead to similar results. As it has been said before, X-Plane will compute the forces acting in these parts and then sum them in order to move the aircraft as a whole.

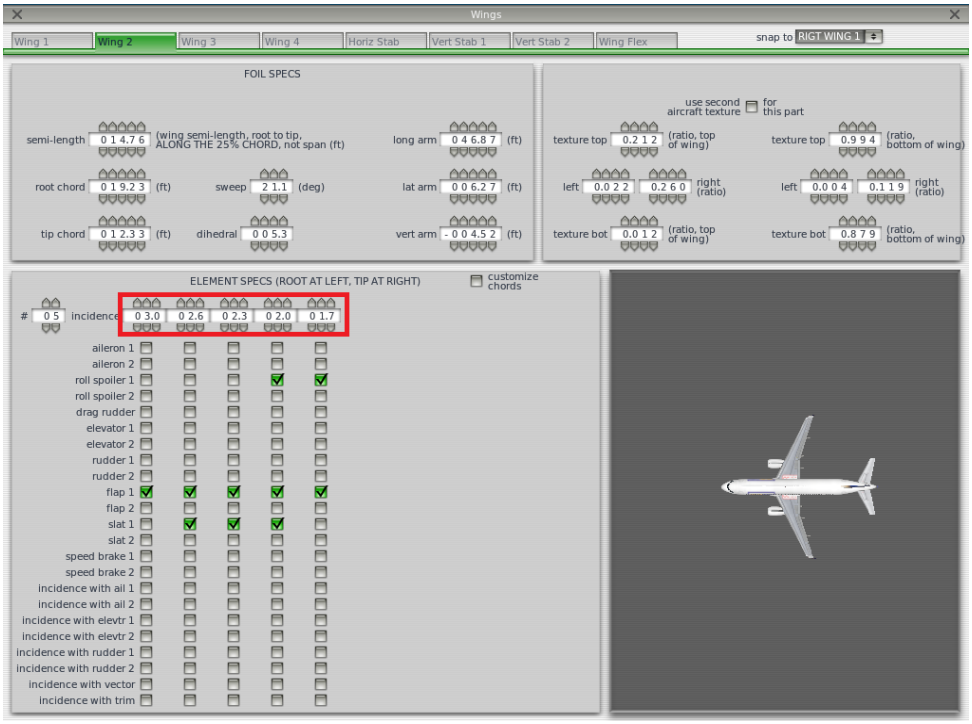


Figure 1.7: Plane Maker wing sections

Once the forces are obtained, it is possible to compute the aircraft accelerations, which are then integrated into velocities and positions.

According to Laminar Research, the use of blade element theory leads to more accurate results than stability derivatives. Most of the simulators nowadays are based on stability and control derivatives [20], which can be explained as follows:

- **Stability derivatives**: they measure how much change happens in a force or moment acting on the aircraft due to a small change in a flight condition parameter, such as altitude, airspeed or angle of attack. They indicate how fast an aircraft returns to straight and level flight for each degree it is offset.
- **Control derivatives**: they measure how much change happens in a force or moment acting on the aircraft due to a small change in the deflection of a control surface such as the rudder, the ailerons or the elevator.

In general, stability and control derivatives are known as aerodynamic derivatives. In table 1.1 it can be seen some examples of aerodynamic derivatives.

Aerodynamic derivative	Nomenclature	Units
Drag coefficient derivative due to angle of attack	CD_a	rad^{-1}
Pitching moment derivative due to elevator deflection	Cm_d	rad^{-1}
Pitching moment derivative due to pitching velocity	Cm_q	rad^{-1}
Lift coefficient derivative due to angle of attack rates	$Cl_{\dot{\alpha}}$	rad^{-1}

Table 1.1: Aerodynamic Derivatives

The problem with using derivatives [5] [6] is the fact that it does not take into account effects such as engine failures, turbulence, stalls, spins, sonic effects or spiraling slipstream, to name a few. Additionally, it is not possible to determine how a new aircraft model will fly. It is necessary to have beforehand a complete understanding of the aircraft performance and stability characteristics. However, if a great amount of data is known for a certain aircraft, the results can be very accurate.

On the other hand, blade element theory figures out the forces acting in different parts of the plane without the necessity of previously known coefficients or aircraft stability characteristics. Although it can be thought a priori that blade element theory is a best option by far, there are also some problems with this method, that will be discussed in the next section.

1.2.2. X-Plane model limitations

X-Plane flight model has some limitations that make it not to be accurate enough in certain situations. For instance, it is not possible to model the fuselage lift and there is a great inaccuracy in the separated flow over surfaces model. In addition, there is a lack of vortex analysis. However, one of the most remarkable limitations of X-Plane is the inaccuracy of transonic and supersonic modeling. Compressible flow effects are considered using Prandtl-Glauert[21], while transonic effects are only simulated by an empirical Mach-divergent drag increase [22]. This is the Mach number at which a great sudden increase

in the drag coefficient happens, and compressible effects start to become apparent. Regarding supersonic flow, the aircraft airfoils are assumed to have a diamond shape with the same thickness ratio as the input airfoils. Pressures behind the shock waves are computed on each of the plates on the diamond airfoil and summed in order to obtain the total pressures on each airfoil element. Additional drag on each component due to supersonic effects is computed by determining the compression shock on each component.

But what is more important for the aim of this project is the transonic modeling and the problems derived from it. Nowadays, it can be assumed that practically all commercial aircraft cruise in the transonic speed range [23], which can be considered to be between Mach numbers from 0.6 to 1.2 approximately. As the Mach number increases, shock waves appear in the flow field, getting stronger as the speed increases. These shock waves make the drag to increase fast, due to both the emergence of wave drag and also the fact that the pressure rise through the shock wave thickens the boundary layer, which means a higher viscous drag. And, as said above, the Mach number associated with this rapid increase in drag is called the drag divergence Mach number, which can be changed in Airfoil Maker.

As a result of this inaccuracy in the transonic model, the behavior of the aircraft at high Mach numbers is not the one expected. There are differences in several values like the lift and drag coefficients and, consequently, in drag and lift forces. Additionally, the angle of attack is not correct. Furthermore, there are some problems with the thrust produced by the engines, and specially with the idle thrust. Apart from all that, it is important to highlight the fact that X-Plane airfoils are only modeled as a function of angle of attack, so there is not a true dependency with the altitude or the Mach number (only the effect of the drag divergence Mach number and the Prandtl-Glauert theory aforementioned). This can be checked in the ".afl" files. An example of these files can be seen in figure 1.8 (from left to tight, Angle of Attack, C_L (lift coefficient), C_D (drag coefficient) and C_m (moment coefficient)). At the top of these files there is also a header with some of the information that can be changed with Airfoil Maker, such as the drag divergence Mach number.

```
-1.5 -0.04600  0.01680 -0.04183
-1.4 -0.03560  0.01680 -0.04173
-1.3 -0.02520  0.01680 -0.04163
-1.2 -0.01480  0.01680 -0.04154
-1.1 -0.00440  0.01680 -0.04144
-1.0  0.00600  0.01680 -0.04135
-0.9  0.01640  0.01680 -0.04125
-0.8  0.02680  0.01680 -0.04115
-0.7  0.03720  0.01680 -0.04106
-0.6  0.04760  0.01680 -0.04096
-0.5  0.05800  0.01680 -0.04087
-0.4  0.06840  0.01680 -0.04077
-0.3  0.07880  0.01680 -0.04067
-0.2  0.08920  0.01680 -0.04058
-0.1  0.09960  0.01680 -0.04048
 0.0  0.11000  0.01680 -0.04038
 0.1  0.12040  0.01680 -0.04029
 0.2  0.13080  0.01680 -0.04019
 0.3  0.14120  0.01680 -0.04010
 0.4  0.15160  0.01680 -0.04000
 0.5  0.16200  0.01680 -0.03990
 0.6  0.17240  0.01680 -0.03981
 0.7  0.18280  0.01680 -0.03971
 0.8  0.19320  0.01680 -0.03962
 0.9  0.20360  0.01680 -0.03952
 1.0  0.21400  0.01680 -0.03942
 1.1  0.22440  0.01680 -0.03933
 1.2  0.23480  0.01680 -0.03923
 1.3  0.24520  0.01680 -0.03913
 1.4  0.25560  0.01680 -0.03904
 1.5  0.26600  0.01680 -0.03894
```

Figure 1.8: Airfoil file body

Another interesting limitation of X-Plane is the fact that if display options are set too high (a high resolution with a lot of details) for the computer being used, the cycle rate for completing calculation will decrease and there is the possibility that aircraft suffers from

abnormal oscillations. Therefore, it is always preferable to set the graphic options to the minimum. As a general rule, it can be assumed that while the frame rate is bigger than 15 frames per second the flight model will work properly and all the calculations will be done.

In Chapter 4, some of the errors explained above will be analyzed more in-depth, quantifying them and solving them.

1.2.3. QPAC A320

The aircraft used in X-Plane to do the simulation tests is the QPAC A320 [24], developed by QualityPark AviationCenter. Some of the features it includes are named below:

- ILS auto-align feature
- Startup Configuration
- PFD and ND synoptic
- Additional ND features
- Improved flight plan editing

For more information about the QPAC A320 the corresponding reference can be consulted. With this aircraft it can be assumed that the systems simulation (fuel, electrical, pneumatic,...) is correct enough for the purposes of this project. Although the QPAC A320 includes some interesting features that improve several aspects of X-Plane, it has to be kept in mind that the flight model is still the same, so the problems commented in section 1.2.2. persist. However, QPAC includes also a plugin (in section 3.1. there is more information of what X-Plane plugins are and how they work) that offers the possibility to change some of the aspects of the flight model, as it will be seen in Chapter 4. This plugin has been developed exclusively for the German Aerospace Center.

In figure 1.9(a) and 1.9(b) it can be seen the QPAC A320 aircraft and cockpit.

(a) QPAC A320 Aircraft

(b) QPAC A320 Cockpit

Figure 1.9: QPAC A320

1.3. Airbus Performance Engineering Program tool (PEP)

Aircraft type identification (which involves the parameter estimation for a particular aircraft model) [25] depends on availability, type and quality of aircraft performance reference data. The main sources of this data in the past were the Aircraft Operation Manuals (AOMs), which are published by aircraft manufacturers or operating airlines. From the point of view of performance modeling, AOMs provide information of aircraft limitations, performances and operating procedures for all aircraft types that have ever been put in operation. Performances are given in form of integrated flight profiles that specify time, distance and fuel to climb/descent to/from specific flight levels. Data is given for number of flight levels, with a variable altitude step which sometimes can give as a result a low number of data points. There are other inaccuracies, such as the fact that the time to climb or descent is rounded to minute. In order to solve these problems, aircraft manufacturers have developed aircraft performance engineering programs that can provide a high quality performance reference data. The advantage of these programs is the fact that they allow generation of reference data for complete range of aircraft operating conditions in terms of weight, speeds, ISA and associated operating regimes with high level of data granularity (a great amount of data points) and data precision.

The Airbus Performance Engineering Program, also known as PEP tool, is a software developed by Airbus that will be used in this project to obtain performance reference data of an Airbus A320. In order to generate the files, it will be used a database provided by Airbus specifically for the Advanced Technology Research Aircraft (ATRA) of the German Aerospace Center (DLR), which is an Airbus A320-232. More information about this aircraft can be found in the section 1.3.1.. DLR also plans to do its own database, generating it from flight tests done with the ATRA, but when this project was done this database was not available yet. PEP tool offers a lot of possibilities, and it allows the user to obtain aircraft performance reference data for several situations and depending on different parameters. It is divided into the following modules[26]:

- **Flight Manual (FM)**: it represents the performance section of the Flight Manual in a digital format.
- **Takeoff and Landing Optimization (TLO)**: takeoff calculation gives the maximum takeoff weight and associated speeds, runway and aircraft atmospheric conditions. The takeoff performance computation is specific to one airframe/engine/brakes combination. The takeoff calculation computes takeoff performance on dry, wet and contaminated runways, taking into account runway characteristics, atmospheric conditions, aircraft configuration (flap setting) and some system failures. Runway and obstacle data have to be provided by the user.
- **Flight Planning (FLIP)**: it allows to obtain fuel predictions for a given air distance under simplified meteorological conditions.
- **In Flight Performance (IFP)**: it computes general aircraft in-flight performance for specific flight phases. It works from the aircraft performance database for the appropriate airframe/engine combination. It can be used to extract digital aircraft performance data to be fed into programs specifically devoted to flight planning computation.

- **Aircraft Performance Monitoring (APM)**: evaluates the aircraft performance level with respect to the manufacturer's book level. It allows the operator to follow performance degradation over time and trigger maintenance actions when required to recover in-flight performance.

- **Operational Flight Path (OFP)**: this module is used to compute the aircraft operational performance. It gives details of all engine performance and also of engine out performance. It allows the operations department to check the aircraft capabilities for flying from or to a given airport, based on operational constraints.

The module that will be used in this project is the IFP, and the performance database used, as it has been already said, will be the one from Airbus for the ATRA. In figure 1.10, it can be seen the PEP tool interface, with the IFP module opened. The different module options are highlighted in red.

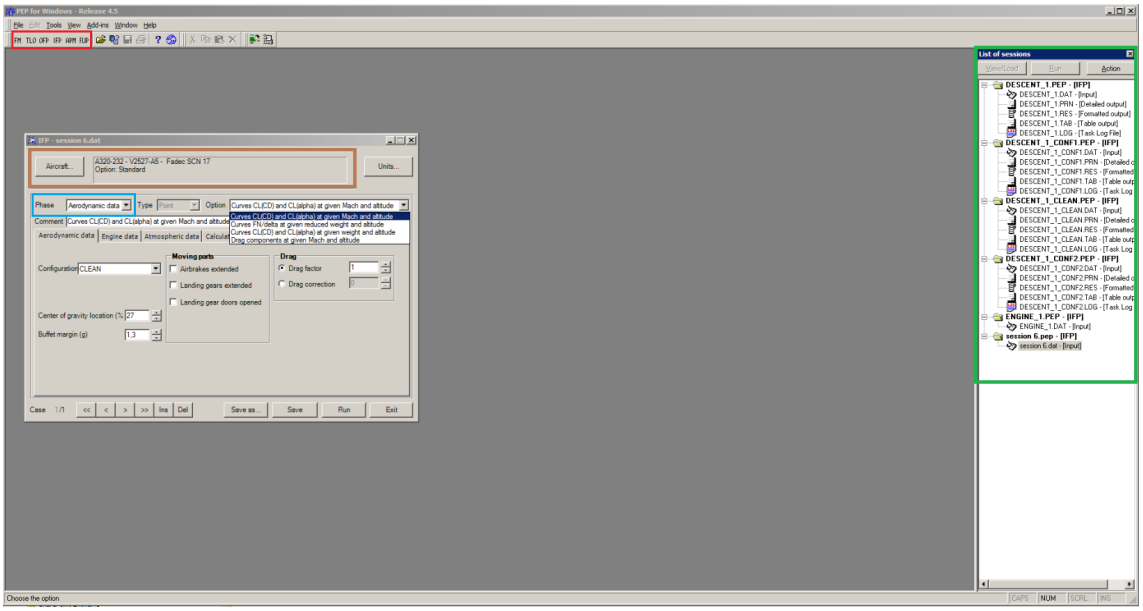


Figure 1.10: PEP tool interface

Highlighted in brown it can be seen a button called "Aircraft...", which is used to select the aircraft database. Next to this button it is found the database, which is called "A320-232 V2527-A5 FADEC SCN 17". Another important concept of PEP tool is the phase tab, highlighted in blue. There, it is possible to select the following phases:

- Aerodynamic Data.
- Engine Data.
- Climb.
- Cruise.
- Descent.
- Descent + Cruise.

- Holding.

It is interesting to remark the fact that each phase has different options, and that it is possible to obtain data depending on parameters such as the weight, altitude, speed, C_L,... In section 2 more details will be given of the options available in some of the phases.

Finally, on the right side, highlighted in green, it can be seen the different sessions done with the IFP module. In this section all the files that the IFP generates for each session are stored. In figure 1.11, it can be seen an example of a part of a file generated by PEP tool. In this case it shows different values for a normal descent at 210 kt. The data is not the real one as it is property of DLR, the figure is only thought to give an idea of the PEP files format.

```
1AIRBUS IFP-V15.4 W  JAN 2009 HSL_V1.0
 A320-232            V2527-A5
 Normal descent at given Mach/CAS law

                                        CONFIGURATION : 1

                                      C G POSITION  :  25.0 %

                                   DRAG FACTOR 1.0000  FUEL FACTOR  1.0000

                                   AVERAGE  ENGINE - FLHV :  18590.BTU/LB

                                        BLEED SELECTION: NORM

                                WITHOUT ANTI-ICING    NORMAL AIR CONDITIONING

                                  WEIGHT   :  65000. KG   ISA +  0.0 DG.C

                                        100.0 % OF IDL  POWER

                                     NO REPRESSURIZATION SEGMENT

                                        DESCENT : 210.00 KT

   ALT.    ALTG    WGHT    MACH    CAS     TAS     WIND    TIME    FUEL    DIST    RATE    GRDT    ALPH     CL      CD     WFE      FN     EPR
 ( FT )  ( FT )  ( KG )  (    )  ( KT )  ( KT )  ( KT )  ( MN )  ( KG )  ( NM )  (FTMN)  (DEG.)  (DEG.)  (    )  (    )  (KG/H)  (DAN )  (    )

  10000.  10000.  65000.   0.381   210.0   243.0     0.0    0.00      0.     0.0  -1021.1   -2.12    5.40 0.72845 0.04312   1025.    793.   0.989
   9000.   9000.  64983.   0.374   210.0   239.4     0.0    0.96     15.     5.9  -1021.9   -2.10    5.39 0.72737 0.04325   1033.    654.   0.976
   8000.   8000.  64966.   0.367   210.0   235.9     0.0    1.92     36.    11.7  -1032.3   -2.76    5.37 0.72653 0.04350   1065.    906.   0.965
   7000.   7000.  64950.   0.360   210.0   232.4     0.0    2.88     58.    13.4  -1041.0   -2.80    5.36 0.42546 0.04312   1048.    555.   0.943
   6000.   6000.  64932.   0.354   210.0   229.0     0.0    3.83     75.    14.0  -1043.7   -2.98    5.35 0.72737 0.04306   1092.    802.   0.943
   5000.   5000.  64915.   0.347   210.0   225.7     0.0    4.78     92.    15.6  -1090.3   -2.99    5.34 0.72737 0.04312   1033.    630.   0.932
   4000.   4000.  64898.   0.341   210.0   222.4     0.0    5.71    111.    19.1  -1065.5   -3.15    5.42 0.72737 0.04325   1054.    521.   0.921
   3000.   3000.  64882.   0.335   210.0   219.2     0.0    6.63    124.    21.5  -1156.2   -3.24    5.29 0.72195 0.04350   1021.    459.   0.945
   2000.   2000.  64866.   0.329   210.0   216.1     0.0    7.53    131.    22.8  -1189.5   -3.63    5.25 0.52737 0.04285   1065.    327.   0.932
   1000.   1000.  64850.   0.323   210.0   213.0     0.0    8.42    172.    34.9  -1243.7   -3.73    5.23 0.62382 0.04350   1079.    196.   0.911
1AIRBUS IFP-V15.4 W  JAN 2009 HSL_V1.0
 INPUT DATA
 END
```

Figure 1.11: Example of an output file of the PEP IFP module

As it can be seen in figure 1.11, the speed for the descent is 210 kt. However, it is possible to generate data for descents at different speeds. The number of these values can be changed in the IFP module, as well as the steps between them. By doing that, it is possible to obtain a great amount of data points, which will lead to more accurate results if there is the necessity to interpolate between points. In this case, it could be of interest to generate data for speeds in increments of 5 kt. And in this particular case it would be also possible to change the amount of altitudes for which the data is available, as well as the altitude steps. All these features make PEP a very powerful tool when it comes to generate accurate data and also with a great amount of data points, which will give as a result a more exact study of the aircraft performance reference data.

1.3.1. Advanced Technology Research Aircraft (ATRA)

The ATRA[27] is an Airbus A320-232, and it was deployed by DLR back in 2008 (see figure 1.12 from reference [27])). It can be understood as a modern and flexible flight test platform which sets a new benchmark for flying test beds in European aerospace research. Some of the research fields of the ATRA are the following ones:

- Testing of aeroelastic measurement techniques.
- Research into inner space acoustics.
- Aircrframe noise measurment.
- Measurement of turbulence (laminarisation) at the wing and the tailplane.
- Measurement of wake vortices.
- Atmospheric and engine measurements.

Figure 1.12: German Aerospace Center Advanced Technology Research Aircraft (ATRA)

During the flight tests it is gathered a lot of data. Some examples were shown in section 1.1. in figure 1.3. With this data, as it has been said before, it will be possible to make in the future a database to use it in the PEP tool, so the data generated with the software will be based on flight tests with the ATRA. So far, however, it is only available an Airbus database specifically for this aircraft.

Some technical data of the ATRA aircraft can be found in table 1.2.

Technical data	**Airbus A320 ”ATRA”**
Length	37.57 meters
Height	11.76 meters
Wingspan	34.10 meters
Cabin length	29.10 meters
Cabin width	3.7 meters
Cabin height	2.4 meters
Seats	maximum 179
Empty weight	42.3 tonnes
Total weight	maximum 75.5 tonnes
Engines	two International Aero Engine V2500 engines
Thrust	111 kilonewton each
Range	4800 kilometers to 5700 kilometers
Flight altitude	maximum 11,800 meters (39,000 feet)
Speed	maximum 840 kilometers per hour
Endurance	up to 2h 30m for test operation
Fuel tank capacity	23,858 liters
Original use	civilian use - passenger aircraft
DLR flight facility	Braunschweig

Table 1.2: ATRA technical data

CHAPTER 2. PROBLEM DEFINITION AND APPROACH

2.1. Performance reference data

As it has been explained in section 1.2.2., X-Plane has some limitations that make its model not to be accurate enough in certain situations. It has to be remarked that the aim of this project is changing the model of the aircraft so that it behaves likes a real one in terms of performance, while the handling qualities remain at a minor interest. Basically, the major aspects that have to be changed are the following:

- Lift model.
- Drag model.
- Thrust model.

Some of the ideas mentioned in section 1 can potentially be a solution to these issues, and in this section it will be discussed what process will be finally followed and why.

In order to be able to change the model the following questions must be answered:

- Which data is needed?
- How will this data be obtained?
- How can the data be introduced in the simulator?

One of the limiting factors of what method can or cannot be used is X-Plane. The simulator allows the user to change certain data, and this is the data that has to be obtained with the method chosen. It is not as easy as saying ”I want a force of 50,000 N when the aircraft is flying at 12,000 ft and 250 kt”.

The first concept explained in section 1 was parameter identification. And together with parameter identification comes the idea of aerodynamic derivatives. One of the possible solutions would be to analyze flight test data in order to extract the aerodynamic parameters and derivatives from it. However, there are some problems with this method that will be listed below:

- Parameter identification is a very time-consuming process. Additionally, it requires a lot of time to learn how to do it properly and also enough experience in this field to obtain good and accurate results.
- Stability derivatives may not be the best solution to the problem, as they have some limitations, as explained in section 1.2.1.. Furthermore, it would be very difficult to include these derivatives in X-Plane.

- At the beginning of the project, when it had to be decided what method should be followed, some of the flight test data needed was not available, so even if some parameters were extracted from it the information would have been incomplete.

Because of the reasons listed above, it was decided not to use parameter identification as a solution to the problem. However, it can be assumed that with this method and enough time and experience it is possible to obtain accurate performance reference data that can be afterwards moved into the simulator. Some look-up tables could be generated, with all the parameters that X-Plane needs. But it is believed that there are faster methods that could lead to better results.

After discarding parameter identification it was decided to check whether PEP tool can fit the needs of this project. With this software, it is possible to generate large tables of data with a big number of data points. As told above, it is wanted to change lift, drag and thrust models. PEP tool allows the user to generate tables of thrust depending on Mach and altitude, with other interesting engine values such as N1 (rotation rate of the low-speed rotor) or EPR (Engine Pressure Ratio), as it will be seen in section 2.2.. Regarding lift and drag, PEP can also generate tables of C_D depending on C_L, Mach, and altitude. Basically, it is possible to generate any kind of performance data needed depending on a large number of parameters. Furthermore, the data obtained can be moved into X-Plane with a plugin, as it will be explained in chapter 3. Therefore, PEP will be the tool used to obtain the performance reference data that will be used after to change the X-Plane performance model. In the next section it will be given more details of the process that will be finally followed in order to change the X-Plane model.

2.2. Performance modeling process

This section is thought to give a general idea of which will be all the steps followed in order to change the performance model of X-Plane. The general process diagram is shown in figure 2.1.

It has to be highlighted that, although X-Plane flight model is based on blade element theory, it has been decided to include some look-up tables like in other existing flight simulators in order to improve the accuracy of the performance model. Therefore, the idea in general terms is using the tables generated in PEP directly in X-Plane, so that via a plugin X-Plane can read the values of the tables and include them in the model. Another solution could have been to do polynomials that for instance could have this form:

$$C_D = A \cdot C_L + B \cdot M \tag{2.1}$$

In equation 2.1 the drag coefficient is computed as a function of lift coefficient and Mach number multiplied by the factors A and B respectively. The problem when using polynomials is that irremediably some accuracy is lost as the correct factors that lead to an exact solution cannot be found most of the time. Therefore, there is always some error. It is better to use look-up tables and if it is needed some value between two data points, it can be interpolated. The accuracy loss will be less. It has also to be thought that the tables generated with PEP have a lot of data granularity, which also reduces the error when interpolating.

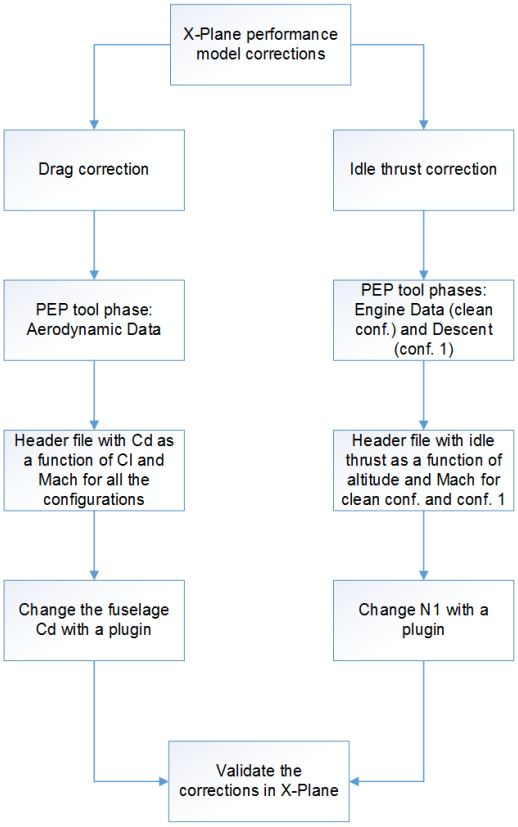


Figure 2.1: Correction process diagram

As it can be seen in figure 2.1, the two corrections that have been finally performed in the simulator are the drag and idle thrust ones. In the next two following sections the process followed to do the corrections will be explained in more detail.

2.2.1. Drag correction steps

It has been found that the easiest way to change the drag model is accessing the fuselage C_D parameter of the aircraft in X-Plane, which can be modified by the user. Therefore, the objective when correcting the drag model is being able to change the fuselage C_D of X-Plane so that the overall C_D of the simulated aircraft becomes the same as the one in PEP tool files. The process followed to do this is the following:

1. Create the PEP tool files with the correct C_D for different values of C_L and Mach. This is the value that is wanted to be obtained in X-Plane after the correction.
2. Generate several text files with the data from the PEP tool files in order to enable a plugin to read the tables easily.
3. Apply the correction. To do it, the fuselage C_D of X-Plane will be changed in such a way that the overall C_D will be the same as the one included in the PEP tool files.

Each step is explained more in detail below:

1. In PEP tool, in the IFP module, it is selected the *Aerodynamic Data* phase. The tables generated are used to obtain the C_D as a function of Mach number and C_L for different configurations (flap/slat). The Mach numbers go from 0.24 to 0.82 in 0.02 increments and the C_L values from 0.2 to 1.2 in 0.04 increments. It has to be remarked that for high Mach numbers some C_L values are not available (the high ones). An example of this table can be seen in figure 2.2. The data is not real as it is property of DLR (the numbers are random), it is only thought to give an idea of the format used.

AIRBUS IFP-V15.4 W JAN 2009 HSL_V1.0
A320-232 V2527-A5
Curves CL(CD) and CL(alpha) at given Mach and altitude

CLEAN CONFIGURATION

C G POSITION : 30.0 %

DRAG FACTOR 1.0000 FUEL FACTOR 1.0000

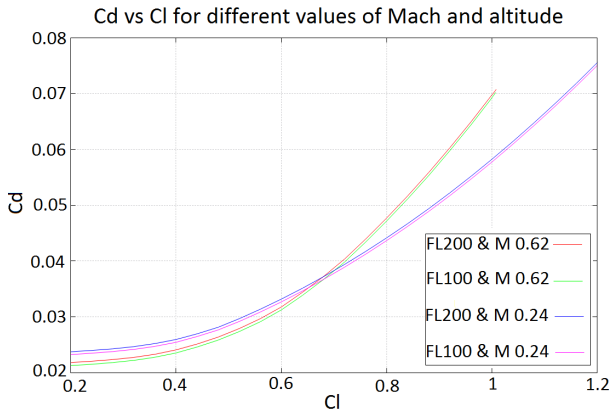
ALTITUDE : 10000. FT ISA + 0.0 DG.C

MACH : 0.240

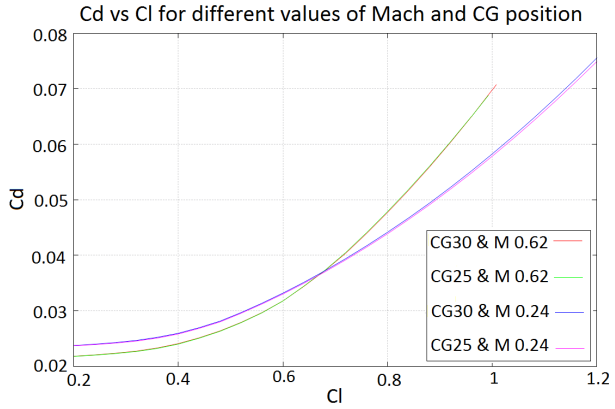
CL ( )	CD ( )	ALPH (DEG.)	LIFT (DAN)	DRAG (DAN)	L/D ( )
0.20000	0.02038	-0.10	6900.8	720.	8.30
0.24000	0.02143	0.40	8265.4	740.	10.02
0.28000	0.02219	0.65	9649.9	760.	11.43
0.32000	0.02312	1.25	11300.5	800.	13.01
0.36000	0.02390	1.78	12560.1	810.	14.13
0.40000	0.02432	2.12	13899.6	832.	15.32
0.44000	0.02500	2.36	15123.2	900.	16.40
0.48000	0.02612	2.98	16156.8	943.	17.21
0.52000	0.02875	3.21	17789.3	1006.	17.45
0.56000	0.03123	3.68	18935.9	1012.	18.08
0.60000	0.03345	4.14	20897.4	1101.	18.10
0.64000	0.03598	4.99	22345.0	1177.	18.23
0.68000	0.03777	5.15	23678.6	1261.	18.77
0.72000	0.03913	5.43	24456.1	1367.	18.65
0.76000	0.04084	6.14	26567.7	1423.	18.32

Figure 2.2: Example of a IFP PEP tool output file to obtain the C_D as a function of C_L and Mach

Furthermore, the altitude and the center of gravity (CG) position for these tables will be always the same: 20,000 feet and 25.0% of the mean aerodynamic chord (MAC) respectively. It is done this way because the effect of these two parameters in the C_D/C_L division is negligible, as it can be seen in figures 2.3(a) and 2.3(b). This fact will make easier the work as it is not necessary to generate more tables, just the ones already mentioned. The difference due to the center of gravity in the C_D vs C_L curves is inappreciable, while the difference due to the altitude is a bit bigger, but still negligible. The Reynolds number is the explanation for the change in the curve due to the altitude. Reynolds affects the drag and depends on the temperature, which depends on the altitude. However, this effect is some orders of magnitude below the effect of Mach and C_L.



(a) Altitude effect on the C_D/C_L curve



(b) CG position effect on the C_D/C_L curve

Figure 2.3: Effect of Altitude and CG on the C_D/C_L curve

Regarding the Mach number effect, the values of the C_D/C_L curve are also quite similar until a Mach number between 0.54-0.62 is reached. In this case, for the same value of C_L the C_D obtained is bigger, as it can be seen in figure 2.4. It is in this Mach number range where the transonic effects start to become apparent, as it has been explained in section 1.2.2.. The transonic effects make the induced drag to increase too, which does it with the square of the C_L. This is the reason why the lines in figure 2.4 corresponding to high Mach numbers have a bigger slope.

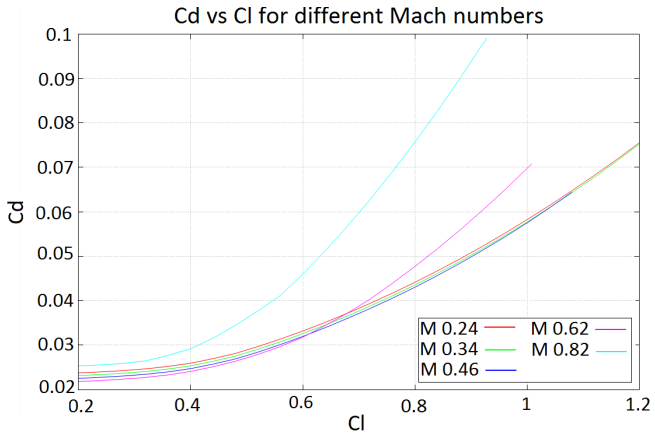


Figure 2.4: Mach effect on the C_D/C_L curve

2. In order to enable the plugin to read the tables easily, the data from them will be extracted with a bash script in Linux. The result from that will be a text file with a big table (one per configuration: Clean, 1, 1+F, 2, 3 and FULL) following the same format like the one shown in table 2.1. Then, this table will be moved into a header file, so that it will be available in the plugin whenever it is needed.

C_L \ $Mach$	0.24	0.26	...	0.82
0.2	C_D values			
0.24				
...				
1.2				

Table 2.1: C_D as a function of C_L and Mach

All the tables generated with PEP tool have been processed with bash scripts in Linux, and more specifically with a tool called *AWK*[28], a pseudo-C interpretor which allows the user to process files very easily.

3. Finally, in order to do the correction of the C_D, the plugin will change the value of the fuselage C_D of the aircraft in X-Plane. The effect of the fuselage C_D on the overall C_D varies with the speed, as it can be seen in figure 2.5. For instance, if the aircraft is flying at 290 kt and it is written in the fuselage C_D a value of 0.1 the overall C_D will be incremented in 0.011, as there is an effect of 11% for a speed of 290 kt.

   In order to generate the figure 2.5, several tests have been done in X-Plane to find the dependencies of the fuselage C_D. After analyzing the effect of different altitudes and speeds, it has been concluded that there is only a relation with the speed. The bigger the speed, the lower the effect of fuselage drag.

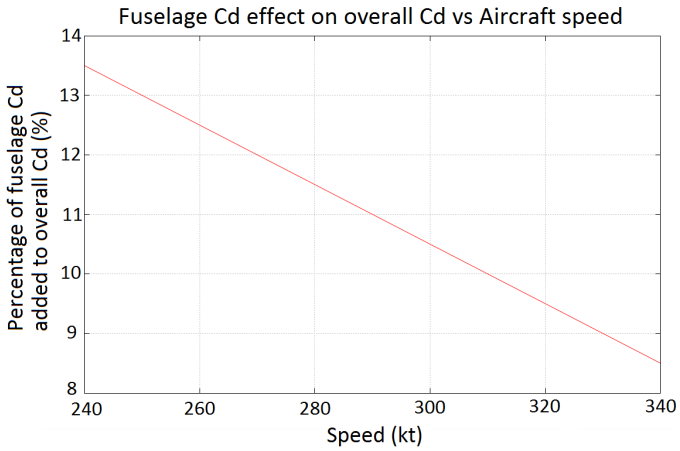


Figure 2.5: Fuselage C_D effect on overall C_D

The relation shown in figure 2.5 is linear, so it is possible to find an equation to obtain the effect of fuselage C_D on overall C_D (I) as a function of speed (V, in knots):

$$I = -0.0000005 \cdot V + 0.000255 \tag{2.2}$$

In equation 2.2, the value obtained is the change produced in overall C_D due to a variation of 0.001 on fuselage C_D. Therefore, in order to obtain the required fuselage C_D the required steps are the following:

$$\Delta_{C_D} = C_{D_{PEP}} - C_{D_{Xplane}} \tag{2.3}$$

$$C_{D_{fuselage}} = 0.001 \cdot (\Delta_{C_D}/I) \tag{2.4}$$

The process followed is simple: the plugin will read from X-Plane the Mach and C_L values in each moment, as well as the current configuration. Then, it will go to the corresponding header file where the tables are located. Most of the times it will be necessary to interpolate between two values of C_L and/or Mach in order to obtain the correct C_D value. Once the correct overall C_D from the tables is obtained, it is computed the necessary fuselage C_D that added to X-Plane will make the overall C_D to be the same as the one in the tables. These computations have been detailed through equations 2.2, 2.3 and 2.4. This process will be done every X-Plane cycle, so it is ensured that the data is always the correct one.

2.2.2. Idle thrust correction steps

In X-Plane, the idle thrust value cannot be written directly in the simulator, as it is not possible to do so. Neither Airplane Maker nor a plugin allow to change the thrust values, so it is needed to find another parameter that controls the thrust. There are two possible thrust setting parameters in an engine [29]. One is N1, which is the rotation rate, in RPM (revolutions per minute), of the low-speed rotor of a two or three-spool engine. Usually, it is expressed as a percentage of some nominal value. The other one is EPR, which is the engine pressure ratio. It is the ratio of total pressure at the exhaust to total pressure at the front of the fan/compressor. If these values are changed, the thrust value also changes. Regarding EPR, it cannot be changed with a plugin, and the possibilities that Airplane Maker offers are not enough to obtain the correct thrust. Therefore, it will be used the N1 parameter. The process followed will involve the next steps:

1. Create the PEP tool files with the correct idle thrust for different values of altitude and Mach. This is the value that is wanted to be obtained in X-Plane after the correction.
2. Generate several text files with the data from the PEP tool files in order to enable a plugin to read the tables easily.
3. Apply the correction. To do it, the N1 value of X-Plane will be changed in such a way that the idle thrust value will be the same as the one included in the PEP tool files.

Now, each step will be explained in more detail:

1. Again, it will be necessary to generate some tables with performance reference data. In this case, two phases of the IFP module in PEP tool will be used: *Engine Data* and *Descent*. The table generated with the phase *Engine data* will be used to obtain

the engine values for a descent with idle thrust in clean configuration, while the one generated with the *Descent* phase will be used to obtain the engine values for a descent with idle thrust in configuration 1. These two tables can be seen in figure 2.6 and 2.7 respectively. Again, the data in both tables is not the real one, as it is only thought to give an idea of the format and the parameters that will be available for the correction.

AIRBUS IFP-V15.4 W JAN 2009 HSL_V1.0
A320-232 V2527-A5
Parameters function of EPR at given speed, alt. and DISA

AVERAGE ENGINE - FLHV : 18590.BTU/LB

BLEED SELECTION: NORM

WITHOUT ANTI-ICING NORMAL AIR CONDITIONING

ALTITUDE : 1000. FT ISA + 0.0 DG.C

THRUST VARIATION FROM MCN THRUST TO IDL THRUST

100.0 % OF MCN THRUST

MACH ( )	TAT (DG.C)	CAS (KT)	TAS (KT)	EPR ( )	FN (DAN)	FG (DAN)	WFE (KG/H)	SFC (KHDN)	EGT (DG.C)	N2 (%)	N1 (%)
0.215	15.66	139.7	141.7	1.310	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.309	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.308	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.276	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.245	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.223	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.123	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.112	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.098	7061.	9473.	3244.	0.452	459.	89.4	81.785
0.215	15.66	139.7	141.7	1.077	7061.	9473.	3244.	0.452	459.	89.4	81.785

Figure 2.6: Engine data for idle thrust descent with Clean Configuration

AIRBUS IFP-V15.4 W JAN 2009 HSL_V1.0
A320-232 V2527-A5
Normal descent at given Mach/CAS law

CONFIGURATION : 1

C G POSITION : 25.0 %

DRAG FACTOR 1.0000 FUEL FACTOR 1.0000

AVERAGE ENGINE - FLHV : 18590.BTU/LB

BLEED SELECTION: NORM

WITHOUT ANTI-ICING NORMAL AIR CONDITIONING

WEIGHT : 65000. KG ISA + 0.0 DG.C

100.0 % OF IDL POWER

NO REPRESSURIZATION SEGMENT

DESCENT : 0. 23 MN

ALT. (FT)	ALTG (FT)	WGHT (KG)	MACH ( )	CAS (KT)	TAS (KT)	WIND (KT)	TIME (MN)	FUEL (KG)	DIST (NM)	RATE (FTMN)	GRDT (DEG.)	ALPH (DEG.)	CL ( )	CD ( )	WFE (KG/H)	FN (DAN)	EPR ( )
10000.	10000.	65000.	0.230	120.0	243.0	0.0	0.00	0.	0.0	-1021.1	-2.12	5.40	0.72845	0.04312	1025.	793.	0.989
9000.	9000.	64983.	0.230	124.0	239.4	0.0	0.96	15.	5.9	-1021.9	-2.10	5.39	0.72737	0.04325	1033.	654.	0.976
8000.	8000.	64966.	0.230	126.0	235.9	0.0	1.92	36.	11.7	-1032.3	-2.76	5.37	0.72653	0.04350	1065.	906.	0.965
7000.	7000.	64950.	0.230	128.0	232.4	0.0	2.88	58.	13.4	-1041.0	-2.80	5.36	0.42546	0.04312	1048.	555.	0.943
6000.	6000.	64932.	0.230	130.0	229.0	0.0	3.83	75.	14.0	-1043.7	-2.98	5.35	0.72737	0.04306	1092.	802.	0.943
5000.	5000.	64915.	0.230	132.0	225.7	0.0	4.78	92.	15.6	-1090.3	-2.99	5.34	0.72737	0.04312	1033.	630.	0.932
4000.	4000.	64898.	0.230	134.0	222.4	0.0	5.71	111.	19.1	-1065.5	-3.15	5.42	0.72737	0.04325	1054.	521.	0.921
3000.	3000.	64882.	0.230	136.0	219.2	0.0	6.63	124.	21.5	-1156.2	-3.24	5.29	0.72195	0.04350	1021.	459.	0.945
2000.	2000.	64866.	0.230	138.0	216.1	0.0	7.53	131.	22.8	-1189.5	-3.63	5.25	0.52737	0.04285	1065.	327.	0.932
1000.	1000.	64850.	0.230	140.0	213.0	0.0	8.42	172.	34.9	-1243.7	-3.73	5.23	0.62382	0.04350	1079.	196.	0.911

AIRBUS IFP-V15.4 W JAN 2009 HSL_V1.0
INPUT DATA
END

Figure 2.7: Engine data for idle thrust descent with Configuration 1

Regarding figure 2.6, not all the values are of interest for the idle thrust correction. It must be noticed the fact that the tables generated show the thrust variation from maximum continuous thrust to idle thrust for different altitudes and Mach numbers. Therefore, only the last row of each table must be included, the one which corresponds to the lowest value of thrust, which is the idle thrust value. The data from clean configuration is available from an altitude of 1,000 ft to 39,000 ft in 1,000 ft

increments (plus an altitude of 39,800 ft) and from a Mach number of 0.23 to 0.81 in 0.02 increments (plus Mach numbers equal to 0.215 and 0.82). These values are thought to cover both the minimum and maximum limitations of altitude and speed of an A320.

Regarding figure 2.7, all the data must be included in the correction. This data is available for altitudes from 1,000 ft to 10,000 ft in 1,000 ft increments and for Mach numbers from 0.23 to 0.43 in 0.02 increments (plus a Mach number of 0.215). As they are tables for configuration 1, it has been considered that a bigger speed will not be achieved by the aircraft, neither a higher altitude than 10,000 ft.

2. As it has been done in the drag correction case, the data from the tables will be extracted with a bash script in Linux so that the plugin can read them easier. The result will be two text files with a table in each of them, like the ones in tables 2.2 and 2.3.

$Altitude(ft)$ / $Mach$	1000	2000	...	39800
0.215	Idle thrust values			
0.23				
0.25				
...				
0.82				

Table 2.2: Idle thrust values as a function of altitude and Mach number for clean configuration

$Altitude(ft)$ / $Mach$	1000	2000	...	10000
0.215	Idle thrust values			
0.23				
0.25				
...				
0.43				

Table 2.3: Idle thrust values as a function of altitude and Mach number for configuration 1

3. Finally, the correction will be applied by changing the value of N1. First, the plugin will go to the corresponding header file depending on the configuration and will read the correct idle thrust depending on the current Mach and altitude of the aircraft. The problem now is that it is not possible to know which N1 corresponds to the idle thrust read from the table, as there is not an equation or a table to find it out. Additionally, it does not work reading N1 from the PEP tables, as in X-Plane the same value of N1 will mean a different idle thrust value than in PEP tool. In order to solve this issue, it has been decided to change the N1 progressively until a minimum error is reached.

   To do this correction, an initial value of N1 is set at the beginning of the simulation, 37%. Then, depending on the difference between the idle thrust from the tables

(which is known for every altitude and Mach) and X-Plane, this value will be incremented or decremented in a given amount. For instance, it could be the case that there is a difference of 2000 N between the idle thrust in X-Plane and the one in the PEP tables (suppose that the idle thrust in X-Plane is bigger). In this situation, the plugin will decrement the N1 value in a 4%. Then, in the next cycle, this difference will be checked again. Imagine that after decrementing the N1 in a 4% now there is difference of 600 N between thrusts. The N1 in this case will be decremented only a 1%. This way the difference between idle thrusts will become smaller and smaller until it is around 0. It is important to remark also that the smaller the difference between idle thrusts, the smaller the change in N1 in order to correct this issue.

There is not a clear mathematical explanation for this, as it has been designed following a trial and error process. By doing that the error cannot be 0, but the results are quite good, as it will be seen in chapter 4. This correction has been possible thanks to the plugin developed by QPAC for DLR, that allows the user to change the value of N1 for idle thrust (more information in chapter 3).

CHAPTER 3. X-PLANE PLUGIN

3.1. Plugin Software Development Kit (SDK)

A plugin [30] is an executable code that runs inside X-Plane, extending what X-Plane does. In addition, they are modular, so they allow developers to extend the simulator without the necessity of having its source code. Plugins are not complete programs in themselves, but rather program fragments that get added to X-Plane while it runs, which allows them to do things that standalone programs might not be able to. Some of the plugin capabilities are listed below:

- Run code from inside the simulator, either continually (for instance, once per flight cycle), or as a response to an event inside the X-Plane.
- Read data from the simulator.
- Write to the simulator.
- Create user interface inside the simulator. For instance, it is possible to create popup windows with instruments or dialog boxes.
- Control various subsystems of the simulator.

Plugins are dynamic linked libraries (DLL). A DLL is a set of compiled function or procedures and their associated variables. They are different from normal programs in some ways. A DLL does not contain a ”main” function that is run once to run the entire program. It contains a lot of functions, and another program uses the DLL by calling these functions. DLL export some functions, which are the *exported* functions, and also have some *internal* functions. External programs can call the exported functions, and in the same way the exported functions can call the internal ones. DLL are powerful as the program that uses them finds and runs the code in the DLL when it is executed, not when it is compiled.

In the case of X-Plane, the central component of the plugin system is the X-Plane Plugin Manager (XPLM), which is a library of code (and also a DLL) that manages plugins. X-Plane links to the XPLM and the plugin links to the XPLM. It is important to highlight the fact that the plugin never communicates directly with X-Plane, it always goes through the XPLM; XPLM also exports the functions that the plugin needs to read or write data, create user interface... Additionally, the XPLM will call the plugin exported functions in order to allow them to modify in several ways X-Plane. XPLM can be understood as a DLL that acts as the central hub for all plugin activity.

When talking about X-Plane plugins it is important to remark the concept of callback functions, which are called by the XPLM to notify the plugin of actions in the simulator, to allow the plugin developer to accomplish the work that the plugin does, or to provide functionality to the user interface. There are two kinds of callbacks: required and registered. The required ones are functions that the plugin exports as a DLL and they are needed to make the plugin load properly. They are listed below:

- XPluginStart: It is called when the plugin is first loaded.

- XPlugin Enable: It is called when the plugin is enabled.
- XPluginDisable: It is called when the plugin is disabled.
- XPluginStop: It is called right before the plugin is unloaded.
- XPluginReceiveMessage: It is called when another plugin or X-Plane sends a message to the plugin.

The registered callbacks are functions that are passed back to the XPLM to access additional simulator functionality. For instance, if a window is created, a registered callback function can be provided, and it will be called when the user clicks in the window.

What is more interesting about the plugin functionalities for the purposes of this project is the fact of reading and writing data. In order to do it, X-Plane introduces the concept of data references[31], more commonly known as *DataRefs*. They are the way the plugin reads X-Plane's internal variables, or exchanges data with another plugin. Datarefs are identified by verbose string names. An example of a dataref is shown below:

$$sim/flightmodel/position/latitude$$

The dataref above stores the value of the aircraft latitude. Datarefs are accessed (read or written) via the XPLMDataAccess.h, an API where several useful functions to access datarefs are stored. The term API[32] stands for "Application Program Interface," though it is sometimes referred to as an "Application Programming Interface." An API is a set of commands, functions, and protocols which programmers can use when building software for a specific operating system. The API allows programmers to use predefined functions to interact with the operating system, instead of writing them from scratch. An example of one of the API functions is shown below:

$$XPLMSetDatai(XPLMDataRef inDataRef, int inValue);$$

In this case, this function is used to write an integer value to a given dataref. In the corresponding reference[33], it can be found a list of all the X-Plane datarefs.

There are some other issues regarding plugins that must be taken into account. First, the plugin and the simulator will not be running at the same time, so if the plugin is slow, the simulator's frame rate will slow down too. If it is fast, it is possible to tell the plugin to apply its changes or functions in every cycle of the simulator. This is the reason why the drag and idle thrust corrections are applied in every cycle of the simulation. However, if a lot of resources were needed by the plugin, it won't be possible to apply the corrections properly. Second, the plugin is in the same process as the simulator, so if the plugin crashes it will crash the whole simulator. Finally, it is not possible to use the X-Plane SDK from another process.

3.2. *XPlaneIO* plugin

The *XPlaneIO* plugin, developed by DLR, is an X-Plane plugin that is used for different purposes. It is organized in several modules, with different functions in each one. For this

project, a new module has been developed in order to do the corrections in the performance model of X-Plane, the *A320FlightModel* module. However, some of the functions that are in the other modules have been used too, so in this section it will be explained both the functions of the *A320FlightModel* module and those modules that were also used in this project.

3.2.1. *A320FlightModel* module

The *A320FlightModel* module is mainly used to do the corrections in the X-Plane performance model. In this section some of the most important parts of the plugin will be explained, in order to have a better understanding of how the changes in the model are done. It has to be taken into account that these corrections will be done every simulation cycle. So they have to be understood as an infinite loop until the flight in X-Plane is finished. The code corresponding to these corrections can be found in annex A. Below it is explained how the idle thrust and drag corrections are done with the plugin:

- For the drag correction, the process followed is shown in the block diagram of figure 3.1.

  It is important to do this correction only when all the flight model is completely loaded. Otherwise, when trying to set the fuselage C_D there are problems with the simulation and X-Plane crashes. In order to solve this issue the correction is only done when the total running time is more than 1 second. This is enough time to allow the simulator to load the whole flight model, which is done once at the beginning of the simulation. It is in this moment when X-Plane breaks the wings, horizontal stabilizers, vertical stabilizers, and propellers (if equipped) down into a finite number of elements to apply after the blade element theory, as explained in section 1.2.1..

  First of all, some initial parameters are obtained. Then, it is necessary to know in which row and column is the correct C_D value, depending on the current C_L and Mach of the aircraft. Most of the times it will be necessary to interpolate, so with the nearest column and row, the C_L, Mach and the table, together with the function *Compute_cd*, it is obtained the correct C_D. After this, it is computed the needed fuselage C_D to obtain the correct overall C_D in X-Plane (it is needed the current speed, together with the C_D of X-Plane and PEP). This is done with the function *Compute_fus_cd* (some of the steps of this function were detailed in section 2.2.). Finally, the fuselage C_D value is moved to the corresponding dataref in X-Plane.

  In order to know which is the current configuration, it is used a dataref that stores the aircraft flap position. For instance, in clean configuration the flap deploy ratio dataref has a value of 0. Furthermore, the drag correction will be applied only when the aircraft is not on the ground. This will happen whenever the load on the gear is less than half times the aircraft weight force.

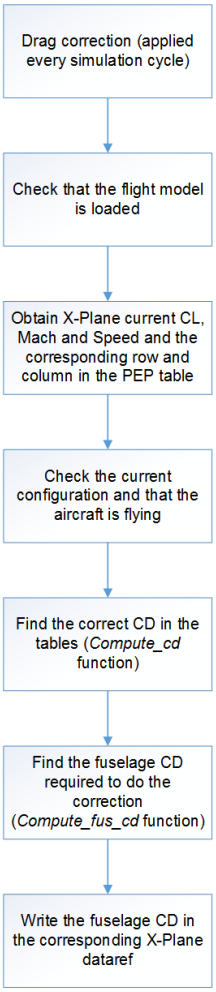


Figure 3.1: Drag correction block diagram

- For the idle thrust correction, the process followed can be seen in the block diagram in figure 3.2.

  In order to extract the correct idle thrust value of the tables, the process followed is like in the drag correction, and the function used in this case is *Compute_idle_conf1* (it requires the Mach, the altitude, the nearest row and column and the table located in the header file). The difference comes when doing the correction. As it has been explained in section 2.2., an initial value of 37% is given to N1, which is stored in the dataref *qpac_airbus_dlrinterface_engoverride_idle_ovrd_value*. Then, this value is incremented or decremented in a certain amount depending on the difference between thrusts of X-Plane and the PEP tables. This amount will be 4%, 1%, 0.25%, 0.1%, 0.08%, 0.06%, 0.03% and 0.01%, which will be added or subtracted from the current N1 in X-Plane depending on the difference between idle thrusts in X-Plane and PEP. The differences considered are $>$1500 N, $>$500 N, $>$200 N, $>$100 N, $>$75 N, $>$50 N, $>$20 N and $>$0 N (in absolute value, and corresponding to each of the amounts of N1 aforementioned). If the X-Plane idle thrust is bigger than the PEP one the N1 will be decremented and in the other case it will be increased. For instance, if there is difference of 220 N between idle thrusts (being X-Plane idle thrust bigger), N1 will be decremented a 0.25%.

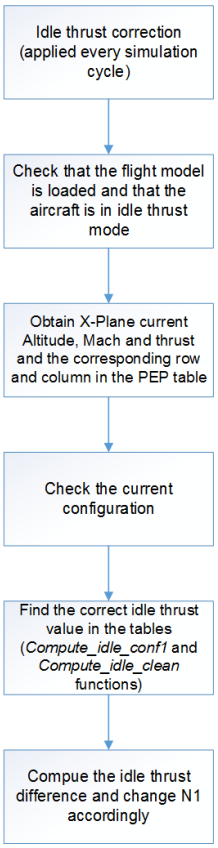


Figure 3.2: Idle thrust correction block diagram

3.2.2. Other plugin modules

Basically, the other plugin functionalities that have been used in this project are located in the modules *QPAC_FBW* and *SimControl*. The first one is the module developed by QPAC for DLR and apart from some new datarefs like the one that allows to change the N1 value it is also used for other things. For instance, whenever the altitude is changed with the autopilot, this plugin is called and it is used to change some of the variables of X-Plane. It was not used directly in this project nor modified in any way, but when using the simulator this plugin is used automatically lots of times.

Regarding the *SimControl* module, it is mainly used to move the aircraft to a given position. The parameters that will be loaded whenever a new position is wanted to be set are the following: altitude, latitude, longitude, pitch, roll, flaps position, gear position, throttle position, track angle, speed and some weather settings. The autopilot will be switched on also. It is very useful when it is needed to do some tests at different altitudes and speeds.

CHAPTER 4. TESTS AND RESULTS

4.1. Experiments performed

After doing the corrections in X-Plane, it is of great interest for the objective of this project to see how the new performance model works now. In order to find it out, several simulations have been done, so that it has been possible to obtain a great amount of data that can be compared with the one in PEP tool tables. The smaller the difference with the PEP values, the better the performance model will be, as it is assumed that the PEP parameters are the correct ones. The simulations can be divided into two groups, depending if they have been performed to obtain drag model data or idle thrust model data. All the simulations have been done with an initial aircraft weight of 64,000 kg, as it is the reference weight specified in the *Operations Performance File* for this aircraft model. This type of files can be found in the Base of Aircraft Data (BADA) by EUROCONTROL[34], which provides a set of ASCII files containing performance and operating procedure coefficients for 338 different aircraft types.

It is remarkable also the fact that the simulations were done with the graphic options set to the minimum. The reason was explained in section 1.2.2.. Therefore, some characteristics of the simulation environment such as the terrain (mountains, cities,...) or weather issues (i.e. clouds) were removed. This way, the simulation goes faster (higher value of frames per second) and possible problems with the behavior of the flight model are avoided.

4.1.1. Drag model simulations

In this case, it has been decided to do three types of descent:

- First descent from 16,000 ft to 1,000 ft at a constant Mach number of 0.44.
- Second descent from 31,000 ft to 12,000 ft at a constant Mach number of 0.62.
- Third descent from 39,500 ft to 25,000 ft at a constant Mach number of 0.82.

Several descents were done for each case. The idea was to check that the behavior for each descent individually was the same always. This way it is possible to ensure that the plugin is working well and there are no random issues that can make it to behave in an unexpected manner. During the three descents, some data was written in an X-Plane output file. The data of interest is the C_L and C_D in each moment and also the difference between the PEP C_D and the X-Plane current C_D. Additionally, these descents were repeated twice, first with the model uncorrected and then with the model corrected. This way, it was possible to generate some graphs to see the difference between both.

The reason why the descents are not the same (same initial and final altitude), is due to the fact that certain Mach numbers are not available in certain altitudes. Therefore, the altitude window has to be limited accordingly. Furthermore, it has been decided to use these Mach numbers and not others because it is believed that by choosing them it can be observed better the effect of Mach number in the curves C_D/C_L; they cover most of the

possible Mach numbers of an A320, and also the point where the transonic effects become apparent, which is of great interest for this project.

Another important issue is the fact that it is very important to keep a constant Mach during all the descent in order to have the right data to compare. As it has been explained in section 2.2., the PEP data available is for a flight level of 200 and a center of gravity of 25% for different Mach numbers and values of C_L. Therefore, it is necessary to have a constant Mach number through all the descent. For instance, in the Mach 0.44 case, if a value of 0.44 is kept constant, it will be possible to compare the values of C_D vs. C_L for this particular Mach number.

All the descents were in real time. It is very slow to do it this way but otherwise the data would not have been correct. The reason is the fact that if X-Plane simulation is accelerated, the data that is being gathered can loose some accuracy. There is not a clear explanation for this, but it is believed that there are some problems when a plugin is working in X-Plane and the simulation is accelerated. Maybe it has something to do with the internal clock of X-Plane and the plugin, some kind of desynchronization between them. The files obtained with an accelerated simulation had several errors in them, with sudden jumps from one value to another that made no sense. Therefore, it was decided to do simulations in real time. Additionally, the data was gathered every 0.1 seconds.

Finally, it has to be highlighted the fact that even when simulating in real time some lines of the text file were removed as they led to data points very far from the correct curve. The reason is that sometimes X-Plane stops for a moment and then carries on. After this ”little crash” the data obtained is slightly different from the expected one (during 2-5 seconds after the ”little crash”), so it has to be removed.

4.1.2. Idle thrust model simulations

For the idle thrust case, two types of descent have been performed:

- First descent from 39,500 ft to 1,000 ft at a speed of 250 kt.
- Second descent from 31,500 ft to 1,000 ft at a speed of 300 kt.

For these two descents, the data gathered was the idle thrust value of X-Plane for the corrected and uncorrected model, and the difference between the X-Plane idle thrust (corrected and uncorrected) and the PEP idle thrust. In this case it was also recorded the PEP idle thrust in every moment, as the tables generated for the idle thrust correction were given as a function of Mach number and altitude, not as a function of speed in knots. In addition, like in the drag case, all the simulations were done in real time (same problems with the accelerated simulations).

The objective here is being able to observe the difference between idle thrusts and also how the speed can affect this difference. In section 4.3., several graphs are shown related with these issues.

4.2. Drag correction results

In section 2.2., in figure 2.4 it has been shown the behavior of the curve C_D/C_L depending on Mach number. In X-Plane these graphs are quite different, as it can be seen in figure 4.1. More or less the same C_D vs C_L is obtained in all Mach numbers, despite the fact that in the higher ones a bigger C_D for the same C_L should be obtained. The slope of all lines is quite similar, while in figure 2.4 there was already a noticeable change in the C_D/C_L curve in Mach 0.62. As the tests were done with X-Plane, it was impossible to get more data points for the graph, so the C_L range varies from approximately 0.23 to 0.55 depending on the Mach number.

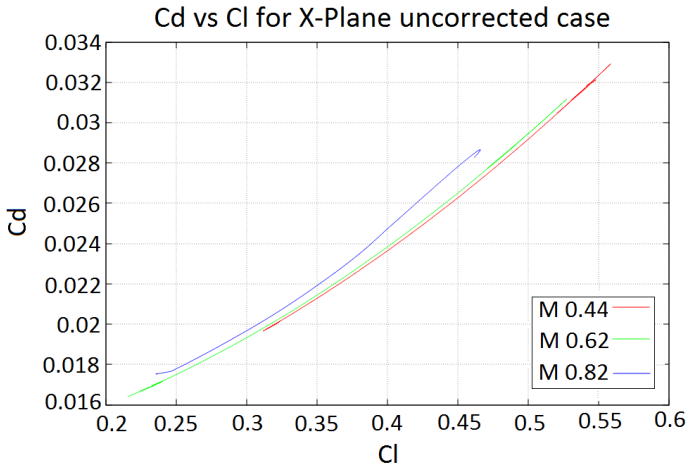


Figure 4.1: C_D vs C_L for X-Plane with no correction

After the drag correction, it can be observed a big improvement in the C_D/C_L curves. The results of the correction can be seen in figures 4.2 and 4.3, corresponding to the C_D/C_L curves for Mach 0.44 and 0.82. As it has been said, there is a great difference between the behavior of the curves in the corrected and uncorrected cases, that becomes greater as the Mach increases.

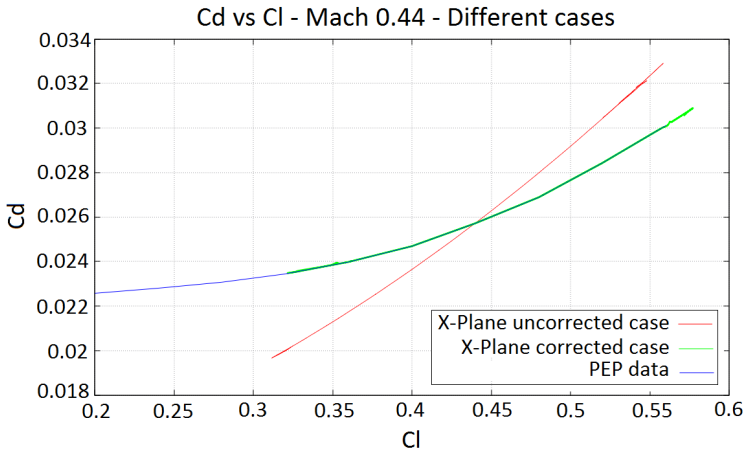


Figure 4.2: C_D vs C_L for X-Plane with no correction

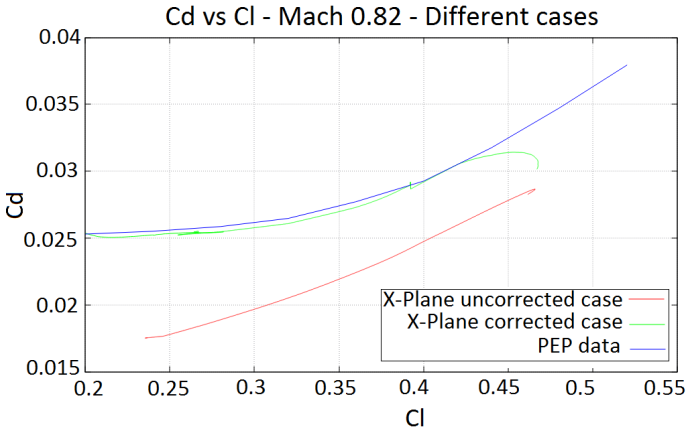


Figure 4.3: C_D vs C_L for X-Plane with no correction

With these graphs it becomes clear the inability of X-Plane to model the drag as a function of Mach number. Neither the use of Prandtl-Galuert nor the divergence Mach number are able to model properly the transonic effects and, therefore, there are inaccuracies with C_D at high Mach. In addition, X-Plane drag model differs from the correct one in low Mach numbers, although the difference is not as big as in higher ones.

In order to obtain the X-Plane data to generate the C_D/C_L curves it was decided to do several descents at constant Mach numbers. This way, the air density changes and it is possible to obtain different C_L and C_D values. However, as it has been said, it is impossible to find more data for higher C_L values due to the aircraft limitations. The autopilot was not able to hold a constant Mach for a higher or lower C_L than the ones shown in the graphs. Additionally, in figure 4.3 it can be seen that the green line, the one corresponding to the corrected case, goes down at the end. This is due to the fact that at the beginning of the descent performed in X-Plane the autopilot was not able to hold a Mach of 0.82, so these values correspond to a Mach between 0.8 and 0.815. Furthermore, this line is always a little bit below the blue line, which is the one corresponding to PEP tool. This is because the Mach was always below 0.82; the autopilot was not capable of keeping this Mach value, and all the descent was made with a Mach of approximately 0.815. The difference is not very big, but there is a noticeable gap between the green and blue lines because of this issue.

Finally, in figure 4.4, it is shown the error in the C_D after the correction. The error is for a Mach number of 0.44 during a descent from 16,000 ft to 1,000 ft, which is the test that was performed to obtain C_D and C_L data for this Mach. Equal or very similar errors can be found for other Mach numbers. It can be assumed an error of 0 in the C_D for all cases. This is a logical solution, as it has been already said that the correction was able to find the exact fuselage C_D that leads to the correct overall C_D, which means an error of 0.

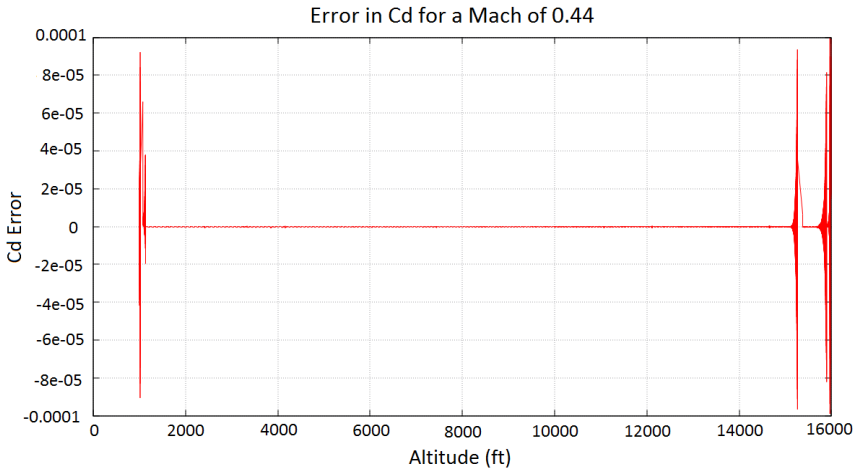


Figure 4.4: Error in C_D with Mach 0.44

4.3. Idle thrust correction results

In order to check the idle thrust correction results it was decided to do a descent in X-Plane at 250 kt from an altitude of 39,800 ft to 1,000 ft. During this test, the idle thrust was recorded and saved in a text file. The test was done with both the corrected thrust and the one without correction, so that it is possible to see which was the initial error and which is the current situation after the correction. In figure 4.5, it can be seen the idle thrust of PEP tool, which is the one correct, with the thrust in X-Plane after applying the correction.

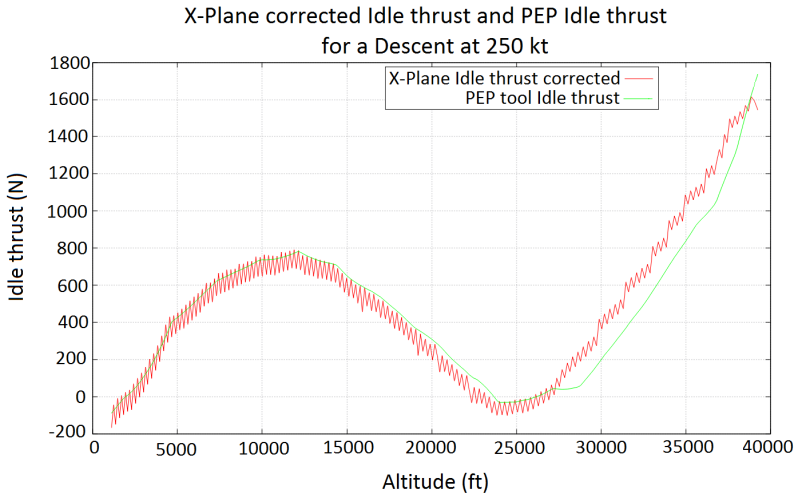


Figure 4.5: X-Plane Idle thrust corrected and PEP tool Idle thrust

It is important to highlight the fact that there are plenty of oscillations in the X-Plane graph because of how the correction is done. As it has been explained in section 2.2., the idle thrust is changed via N1. But the problem is that N1 is incremented or decremented every cycle in a given amount, which has an approximated value found after some trial and error tests with the simulator. The N1 obtained, thus, will lead to an approximated idle thrust, which will change in every cycle as the plugin tries to get the right value. However, the result obtained after the correction is quite good. This becomes clearer if we compare this graph with the X-Plane original idle thrust, without the correction, as it can be seen in

figure 4.6.

X-Plane thrust model gives as a result more thrust at lower altitudes and less at higher altitudes. In addition, the values are very high for idle mode. While the correct idle thrust is always quite low, around 0, with a maximum value of approximately 1,700 N at 39,800 ft, with X-Plane it is possible to obtain a value as high as 10,000 N at an altitude of 1,000 ft.

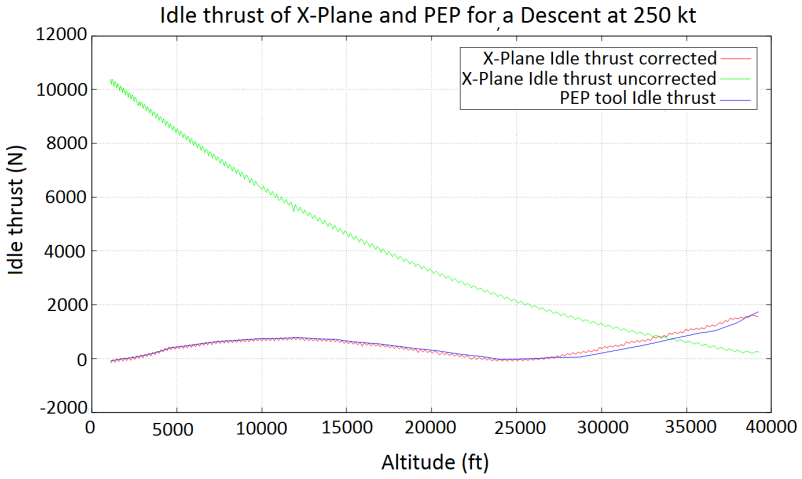


Figure 4.6: Idle thrust of X-Plane (with and without correction) and PEP tool

Finally, in figure 4.7, it is possible to see which is the error in thrust depending on the altitude with and without correction. It has been decided to include two vertical axis in order to have a better understanding of the error behavior for both cases. In the corrected thrust case a greater error is found at higher altitudes, while in the uncorrected case it is the other way around. However, the error is far bigger in the uncorrected case, with a maximum of approximately 10,000 N at an altitude of 1,000 ft. The maximum error in the corrected case is approximately 250 N, and from more or less 25,000 ft and below the idle thrust oscillates in such a way that the error is always around 0.

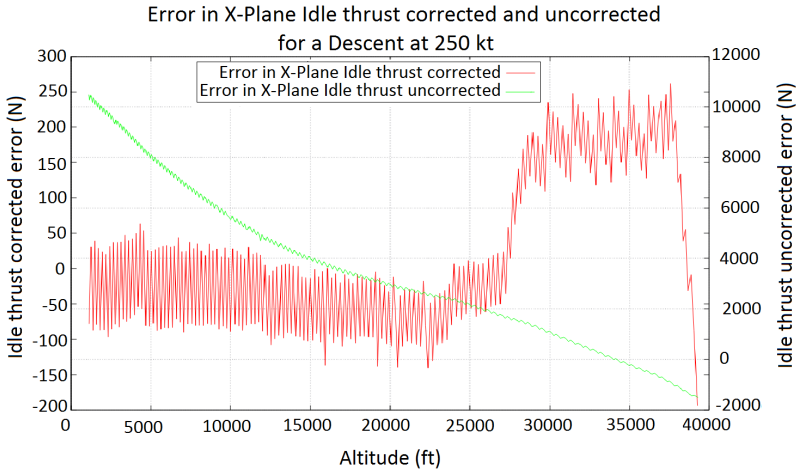


Figure 4.7: Error in Idle thrust for X-Plane with and without correction

Although the results obtained are very satisfactory, it is believed that it is possible to find a more accurate correction. Basically, it will be interesting to study the effect of the altitude

in the thrust model, so that the correction in N1 is done depending on different ranges of altitudes. But as it has been told it is not trivial to find the correct amounts to change N1, as they are obtained after a lot of tests until reaching a good solution.

Regarding the speed, however, the effect is lower. Some other tests with different speeds were also done in order to ensure the same behavior like in the Descent at 250 kt case. The results were quite similar, but with some minor differences. This issue can be seen in figure 4.8. It is shown the comparison in errors of idle corrected thrust for a descent at 250 kt and 300 kt. As it can be seen, the maximum error is approximately the same in both cases. The most noticeable issue is the fact that the altitude seems to have a different effect in the graphs. However, as the error range is similar in both cases, it can be assumed that the same correction method can be applied in all the speeds.

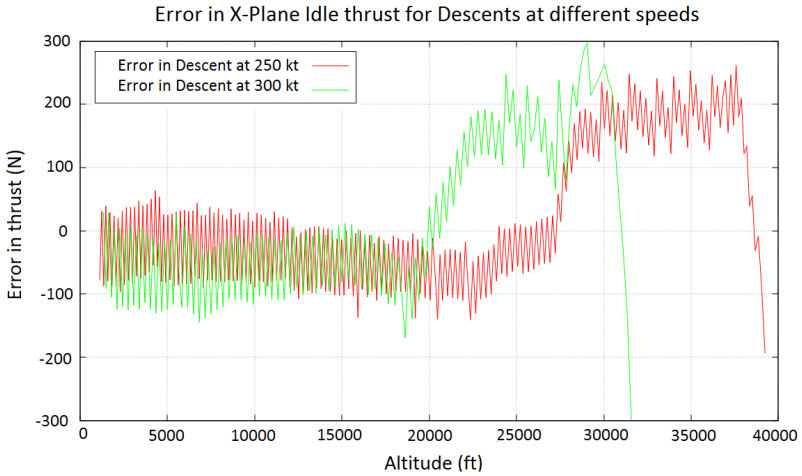


Figure 4.8: Error in Idle thrust for X-Plane with and without correction (300 kt case)

Finally, it is important to remark that in all cases the error at the beginning of the simulation (at the highest altitude) is quite big due to the fact that an arbitrary value of N1 is set, 37%. This value usually leads to a not very accurate value of Idle thrust, but in one or two cycles a better N1 value is found and the error goes down.

4.4. Remaining problems and possible solutions

After doing the corrections, it can be assumed that now both the idle thrust and drag models are correct. However, there are other parameters that are still incorrect. The angle of attack of the aircraft is not the one that can be found in PEP tables, it differs from it approximately in 0.5 degrees with some variations with the altitude. Additionally, the thrust produced by the engines in cruise is not correct neither. Regarding the cruise thrust case, it has to be remarked the fact that it is not possible to use N1 like in the idle thrust correction, as this N1 is exclusively used for the case when the aircraft is in idle (QPAC plugin limitations). One of the possible solutions to solve these issues could be using some new datarefs included in the last version of X-Plane[35], which allow the user to apply arbitrary forces and moments to the aircraft. There are three force datarefs (units in Newtons), corresponding to the X, Y and Z axis of the aircraft respectively:

$$sim/flightmodel/forces/fside\_plug\_acf$$
$$sim/flightmodel/forces/fnrml\_plug\_acf$$
$$sim/flightmodel/forces/faxil\_plug\_acf$$

Also, there are three rotational moments datarefs (units in newton-meters). L is a roll moment (positive s right roll), M is pitch (positive is nose up) and N is yaw (positive is yaw clockwise from above the arcraft).

$$sim/flightmodel/forces/L\_plug\_acf$$
$$sim/flightmodel/forces/M\_plug\_acf$$
$$sim/flightmodel/forces/N\_plug\_acf$$

It is possible to apply forces in arbitrary locations of the aircraft using the moments values. Given a force vector Fx, Fy, Fz, applied at a location X, Y, Z, the additions of these six datarefs are the following:

$$fside+ = Fx; \quad L+ = Fx \cdot Y - Fy \cdot X;$$
$$fnrml+ = Fy; \quad M+ = Fz \cdot Y - Fy \cdot Z;$$
$$faxil+ = Fz; \quad N+ = Fz \cdot X – Fx \cdot Z;$$

Some tests have been done in X-Plane with different values on these datarefs to check their effect on other parameters. It has been observed that approximately a constant change on pitch moment causes a constant change in the angle of attack, so this fact may be a solution to the inaccuracies in the alpha modeling. In addition, it would be possible to add some Z extra force to the aircraft to compensate for the errors in cruise thrust. However, more research should be done on these issues, as for instance changing the angle of attack will also have an effect on drag, so it has to be checked whether these changes are compatible with the drag and idle thrust corrections or not. The process to do this kind of corrections will be the same as the one followed with the drag and idle thrust: obtain PEP tables, read the required values from them and apply the needed changes in X-Plane via these new datarefs. In order to study the effect of these datarefs more easily, some changes have been added to the QPAC A320 with Plane Maker. Basically, six knobs have been included in the cockpit, each of them with a screen next to it. This way, it is possible to change the forces and moments while flying and check at the same time the effect of these changes in the simulation.

Apart from the datarefs, it would be interesting to make also a further study of both Airfoil and Plane Maker. These two software allow the user to change different aspects of the aircraft and, although in this project they have not been used in the end, it may be useful to change other things of the performance model.

CHAPTER 5. APPLICATION ON CONTINUOUS DESCENT OPERATIONS (CDO) TRAJECTORIES

5.1. Concepts of CDO and 4D CDO

Currently, due to the increasing worldwide air travel [4], the existing Air Traffic Management (ATM) infrastructure is very overloaded, as well as the situation in some airports, which are very congested. In addition, the environmental impact is also an increasing problem and is becoming a limiting factor for the air transport stakeholders. Both the Single European Sky ATM Research SESAR and the US Next Generation Air Traffic System (NextGen) have described the existing problems and have identified the need of research and further development of new technologies and aircraft operations that could lead to a more efficient use of the available capacity of the airspace.

In the context of the development of new aircraft operations, it can be highlighted the concept of CDO, which offers a very viable alternative to the usual step down approaches. Nowadays, air traffic controllers use speed and altitude instructions to separate approaching and departing flow, causing the need of a level segment for an extended period of time. This level segment sometimes is performed at a low altitude, causing noise problems and the need of constantly burning fuel and generating gaseous emissions. Furthermore, the current procedure does not adjust to the optimal behavior characteristic for each aircraft, resulting in a more standard operation that does not optimize each aircraft's resources and performance capabilities.

As defined by the International Civil Aviation Organization a CDO is "*An aircraft operating technique aided by appropriate airspace and procedure design and appropriate air traffic control (ATC) clearances enabling the execution of a flight profile optimized to the operating capability of the aircraft, with low engine thrust settings and, where possible, a low drag configuration, thereby reducing fuel burn and emissions during descent*". The shape of the vertical profile follows a continuous descent path that maintains the airplane at higher altitudes than the stepped-down approach and as a result, decreases the noise pollution in the airports nearby. Fuel consumption is also reduced as a consequence of the low thrust setting (ideally on idle) as well as the emission of contaminants. In figure 5.1 it is shown the difference between the CDO and stepped-down approach vertical profiles.

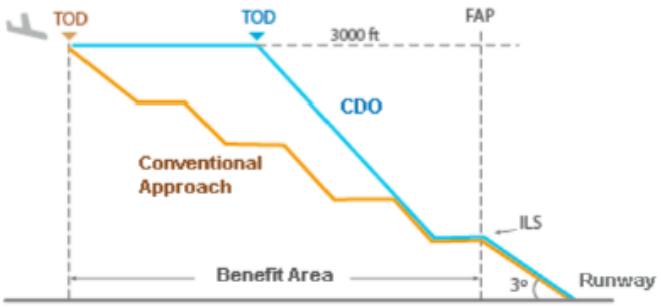


Figure 5.1: Comparison of CDO and stepped-down vertical profiles (reference [4])

Current CDO profiles are calculated with the Vertical Navigation (VNAV) function of the

existing Flight Management Systems (FMS). The descent profile is calculated backwards from the runway threshold towards the Top of Descent (ToD). Apart from the conventional concept of CDO, it has been developed also the concept of 4D CDO, which involves adding time constraints in CDO trajectories. For more information about CDOs, it can be consulted reference [4].

5.2. DLR CDO simulations results

DLR is currently testing 4D CDO via several simulations, where X-Plane is used as the flight simulator. Besides, for displaying the FMS planned trajectory, there is an additional software developed by DLR which gives an improved view of the aircraft's altitude, speed, thrust profile and other characteristics of the aircraft. Furthermore, the time and altitude errors of the aircraft every moment are shown in a real-time graph. The different colors represent different sub-phases of the flight that have been used while planning the initial altitude and CAS profile. In figure 5.2, it is shown this software with the initially planned altitude and speed profiles.

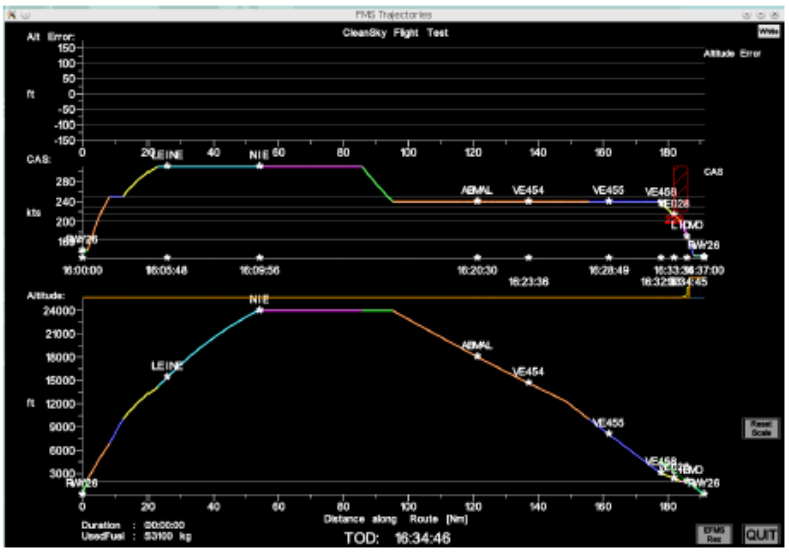


Figure 5.2: Initially planned altitude and speed profile ([4])

The objective of DLR is being able to test the CDO trajectories successfully and with enough accuracy so that they can use the results from the simulations for further studies of the 4D CDO concept. However, these simulations depend, first, on the flight simulator being used. In this case it is X-Plane, which means that its performance model limitations will affect in a negative way the output of the CDO simulations. An example of this issue is shown in figure 5.3, where it can be seen the results of a simulation with no wind in the runway (RWY) 26 of EDVE (Braunschweig Airport). As it can be observed, the actual trajectory followed by the aircraft does not match the one previously planned by the FMS. It tends to go below it. Additionally, although in CDO trajectories is preferable to do the descent in idle thrust, in this simulation this condition is not met. Whenever the actual path goes below a certain altitude respect to the planned path, a speed command is sent to the aircraft so that it can come back to the real path (so the descent is not in idle thrust). The result of this issue are some "bounces": the aircraft starts to go below the planned

trajectory until a speed command is given to go back to it; the process is repeated through all the trajectory. And all these problems are happening in a situation with zero wind, where less inaccuracies are supposed.

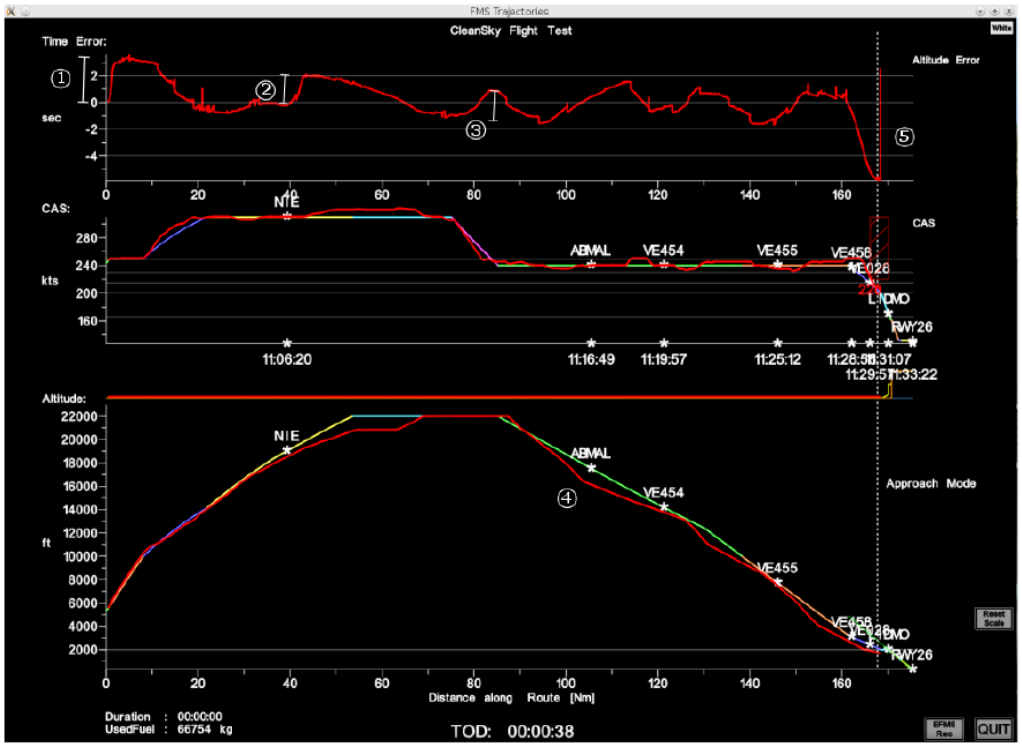


Figure 5.3: CDO simulation for RWY26 of EDVE - No wind (reference [4])

It can be assumed that, among other possible reasons, X-Plane is responsible for, at least, part of this aircraft incorrect behavior. A performance model with inaccuracies in both drag and thrust (in this case idle thrust is the part of interest) models can cause a great amount of errors in a simulation of a CDO trajectory. In next section it will be commented how improving the X-Plane performance model would lead to better results in the simulations.

5.3. Better simulation results with X-Plane improved performance model

In order to know how the improvement of the performance model of X-Plane could improve the results of the simulations two descents have been performed in the flight simulator. Both were from FL320 to 2500 ft MSL (Mean Sea Level), but one with the corrected performance model and the other one with the uncorrected one. From FL320 to 4000 ft MSL the speed was 250 knots (constant Calibrated Airspeed) and in 4000 ft MSL a deceleration to 217 knots (green dot speed) was performed. The results of those descents are shown in figure 5.4. As it can be seen, the line corresponding to the descent with the corrected model (the magenta one) is above the line corresponding to the uncorrected model, where the descent is steeper. If this behavior is moved to the CDO trajectories simulation, it can be assumed that at least part of the error could possibly be removed, leading to more accurate results. However, due to a lack of time, it was not possible to do the CDO

simulations.

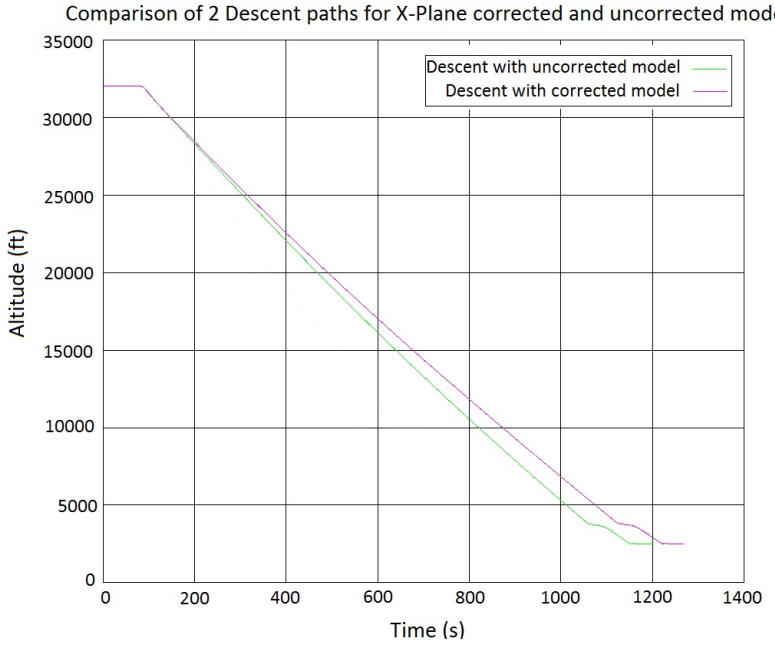


Figure 5.4: Comparison of 2 Descent pahts with X-Plane corrected and uncorrected model

CONCLUSIONS

The main objective of this project was changing the performance model of X-Plane and it can be said that the results obtained are very satisfactory. Now, after correcting the drag and idle thrust models, it is possible to do more accurate simulations with this software, which will allow to test several aeronautical concepts that otherwise it would have not been possible to test successfully.

The process followed to improve the performance model of X-Plane includes the use of PEP (Performance Engineering Program tool) tables with the a database created by Airbus for the ATRA (Advanced Technology Research Aircraft) in DLR. However, in the future it will be possible to do the correction also with PEP but with a database obtained from flight tests in DLR. This will probably mean a different model, and maybe even better results.

It is important to remark also the fact that, even after changing the drag and idle thrust models, there are still some aspects that can be improved of the performance model. For instance, neither the angle of attack nor the thrust in cruise are correct, so more research has to be done on this issue to correct it. Regarding the corrections already done, there is also some margin of improvement. Although the error is quite low in both cases, it is believed that a better correction can be done in the case of the idle thrust, choosing better values of N1 that will mean more accurate values of idle thrust.

Apart from the use of PEP, other ways could have been followed in this project to solve the same problem. One of them, also explained here, is the use of parameter identification from flight test data. With this method and the necessary flight test data it would be possible to generate also some kind of look-up tables for X-Plane that can be moved to the simulator via a plugin. However, when trying to change the performance model of X-Plane it is always very important to keep in mind that not all the data is suitable to be written into the software. This is one of the most important limiting factors when doing the corrections: finding the right parameters that X-Plane allows the user to change, either directly (like in the drag coefficient case) or indirectly (like in the idle thrust case). Furthermore, a more in-depth study should be conducted on how X-Plane works. There are several aspects of the flight model that are unknown for the user, as not a lot of information is provided.

It can also be concluded that once some knowledge of plugin development is acquired, X-Plane is a very powerful simulator in terms of customization. X-Plane plugins allow to read and write a lot of data, and that is one of the most interesting features that X-Plane offers. This way it has been possible to change the performance model but also, when doing flight simulations, there is the possibility to study the effect of the different tests performed in the simulator parameters, just by writing them into an output file while in simulation for a later study. Apart from the plugins, there are also Plane Maker and Airfoil Maker, which add even more customization options to the software.

Overall, it can be assumed that most part of the project objective has been achieved with good results. X-Plane has proven to be a really suitable option to change the performance model and with more research on all its features it is believed that even more improvements can be done. This project can be understood as a starting point towards the complete improvement of the performance model of X-Plane, by establishing some of the required steps to do it.

BIBLIOGRAPHY

[1] P.G. Hamel and R.V. Jategaonkar. Evolution of flight vehicle system identification. *Journal of Aircraft*, 33(1), 1996. DLR, German Aerospace Research Establishment, Braunschweig, Germany. ix, 3, 4

[2] K.W Iliff and K.C. Wang. Retrospective and recent examples of aircraft parameter identification at NASA Dryden Light Research Center. *Journal of Aircraft*, 41(4), 2004. ix, 6, 7

[3] Aerodynamics for Students. Analysis of propellers. http://www-mdp.eng.cam.ac.uk/web/library/enginfo/aerothermal_dvd_only/aero/propeller/prop1.html, 2015. [Online; accessed 10-June-2015]. ix, 8, 9

[4] B. Bendris. *Treball de Fi de Grau - Flight Test Preparation of a 4D-controller for time constrained Continuous Descent Operations (CDO)*. UPC (Polytechnical University of Catalonia) and DLR (German Aerospace Centre), 2015. ix, 43, 44, 45

[5] C.W.S. Thong. *Modeling Aircraft Performance and Stability on X-Plane*. University of New South Wales at the Australian Defence Force Academy, Australia, 2010. 1, 10

[6] D.W. Babbka. *Flight Testing in a Simulation Based Environment*. California Polytechnic University, San Luis Obispo, California, 2011. 1, 10

[7] Several authors. X-Plane.Org Forum. http://forums.x-plane.org/. [Online; accessed 2-August-2015]. 1

[8] Several authors. X-Pilot.com. http://forums.x-pilot.com/page/index.html. [Online; accessed 2-August-2015]. 1

[9] R.V. Jategaonkar. *Aerodynamic Modeling and System Identification from Flight Data-Recent Applications at DLR*. Ankara, Turkey, 2008. 3

[10] Q. Chu, D. Choukroun, C. de Visser, G. Looye, B. Mulder, and E.J. van Kampen. *Advances in Aerospace Guidance, Navigation and Control*. Berlin, 2013. 4

[11] Institute of Flight System Dynamics Tecnishe Universität Müunchen. Simulation, Parameter Identification and Flight Safety. http://www.fsd.mw.tum.de/research/modeling/, 2014. [Online; accessed 14-June-2015]. 5

[12] G. Chowdhary and R.V. Jategaonkar. *Aerodynamic parameter estimation from flight data applying extended and unscented Kalman filter*, 2010. 6

[13] K.W Iliff and R.E. Maine. *Bibliography for Aircraft Parameter Estimation*. NASA, USA, 1986. 7

[14] Laminar Research. *Plane Maker App for X-Plane 10 Manual*, 2013. 8

[15] Laminar Research. *X-Plane 10 Manual*, 2014. 8

[16] L. Preisner. Blade element and momentum theory. http://www.helis.com/howflies/bet.php, 1997. [Online; accessed 10-June-2015]. 8

[17] S. Drzewiecki. *Des Hélices propulsives*. Paris, 1900. 8

[18] S. Drzewiecki. *Du Choix des elements determinant les hélices propulsives permettant leur facile comparaison entre elles*. Paris, 1901. 8

[19] Laminar Research. Appendix A: How X-Plane Works. http://wiki.x-plane.com/Appendix_A:_How_X-Plane_Works, 2011. [Online; accessed 10-June-2015]. 9

[20] H. Loewen. *Stability Derivatives: What they are and how they are used*. MicroPilot, Canada, 2013. 10

[21] AeroStudents. Subsonic compressible flow over airfoils. http://aerostudents.com/files/aerodynamicsC/subsonicCompressibleFlowOverAirfoils.pdf. [Online; accessed 11-June-2015]. 10

[22] The University of Manchester, Manchester. *MATH45111: Compressible and Incompressible Fluid Dynamics. Lecture 24.* 10

[23] W.H. Mason. *Transonics Aerodynamics of Airfoils and Wings*, 2006. 11

[24] QualityPark AviationCenter GmbH (QPAC). http://www.qpac-us.com, 2015. [Online; accessed 12-June-2015]. 12

[25] A. Nuic, C. Poinsot, M.G. Iagaru, E. Gallo, F.A. Navarro, and C. Querejeta. *Advanced Aircraft Performance Modeling for ATM: Enhancements to the BADA Model*. Eurocontrol Experimental Centre and Boeing Research and Technology Europe, 2005. 13

[26] Flight Operations Support and Services. *Getting to grips with A320 Family performance retention and fuel savings*. Airbus, 2008. 13

[27] German Aerospace Center (DLR). Airbus A320-232 "D-ATRA". http://www.dlr.de/dlr/en/desktopdefault.aspx/tabid-10203/339_read-277#/gallery/110, 2012. [Online; accessed 22-June-2015]. 16

[28] Bruce Barnett and General Electric Company. Awk. http://www.grymoire.com/Unix/Awk.html#uh-23, 2001. [Online; accessed 12-May-2015]. 24

[29] Flight Operations Engineering. *Propulsion (1) Jet Engine Basics*. Boeing, 2012. 25

[30] B. Supnik. X-Plane SDK Documentation-Overview. http://www.xsquawkbox.net/xpsdk/mediawiki/Overview, 2015. [Online; accessed 24-June-2015]. 29

[31] B. Supnik. X-Plane SDK - Documentation-Datarefs. http://www.xsquawkbox.net/xpsdk/mediawiki/DataRefs, 2010. [Online; accessed 24-June-2015]. 30

[32] TechTerms. API. http://techterms.com/definition/api, 2015. [Online; accessed 25-June-2015]. 30

[33] B. Supnik. X-Plane SDK - Datarefs for X-Plane 10.40. http://www.xsquawkbox.net/xpsdk/docs/DataRefs.html, 2015. [Online; accessed 24-June-2015]. 30

[34] European Organisation for the Safety of Air Navigation. *User manual for the base of aircraft data (BADA) Revision 3.9*. EUROCONTROL, 20011. 35

[35] B. Supnik. X-Plane SDK - MovingThePlane. http://www.xsquawkbox.net/xpsdk/mediawiki/MovingThePlane, 2014. [Online; accessed 8-August-2015]. 41

APPENDICES

APPENDIX A. *A320FLIGHTMODEL* MODULE CODE

In figure A.1, it can be seen the main code corresponding to the correction of the drag model. The figure corresponds to the case of clean configuration.

```
//DRAG CORRECTION

if(SimData->time_total_running_time_sec>1.)
{
    mach_no=SimData->flightmodel_misc_machno;
    altitude=(SimData->flightmodel_position_elevation)/0.3048;
    speed=SimData->flightmodel_position_true_airspeed*0.51444444;
    speed_ias=SimData->flightmodel_position_indicated_airspeed;

    cl1=SimData->flightmodel_misc_cl_overall;


    row=int((cl1*25)-4);
    column=int((mach_no*50)-11);

    if((fabs(SimData->flightmodel2_controls_flap_handle_deploy_ratio-0)<=0.001)
       &&(SimData->flightmodel_forces_fnrml_gear > (SimData->aircraft_weight_acf_m_empty * 9.81 * 0.5)))
    {
        cd=Compute_cd(cl1, mach_no, row, column, aero_clean_table);
        double fuselage_cd=Compute_fus_cd(speed_ias, cd, SimData->flightmodel_misc_cd_overall, A320FlightModel.fus_cd);


        A320FlightModel.fus_cd=fuselage_cd;


        if((cd-0)<=0.000001)
        {
            printf("There is no data for this cl and mach!");
        }

    }
```

Figure A.1: Drag correction code

In figures A.2 (for configuration 1) and A.3 (correction done when the idle thrust in X-Plane is bigger than the one in PEP tables), it can be seen the main code corresponding to the correction of the idle thrust. It is shown the case where the idle thrust in X-Plane is bigger than the one in PEP tables, so the N1 is decremented.

```
//THRUST CORRECTION
//The current thrust of X-Plane is obtained here
memcpy(&Thrust[0],&Thrust[1],99*sizeof(float));
Thrust[99] = SimData->flightmodel_engine_POINT_thrust[0];
int i;
float PWR=0;
for(i=0;i<100;i++)
{
    PWR=PWR+Thrust[i];
}
PWR=PWR/100.;

//If the IDLTHR_Enabled is true the correction will be done; also, it is rquired that the autothrust mode is 4, which means idle thrust.
if((SimData->sim_time - last_upd_thr) > 5.0 && SimIO_Settings["A320FlightModel"].getBool("IDLTHR_Enabled"))
{
    if(pA3MSData.qpac_airbus_pfdoutputs_general_athr_thrust_mode==4)
    {
        pA3MSData.qpac_airbus_dlrinterface_engoverride_enable_idle_ovrd = 1;
        if(pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value < 1.0)
            pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value = 37.0;

        float RPWR = 0.;

        //Reading a value on the table (depending on the configuration) to correct the thrust.
        mach_no=SimData->flightmodel_misc_machno;
        altitude=(SimData->flightmodel_position_elevation)/0.3048;

        row=int((mach_no)*50-9.5);
        column=int((altitude)/1000);

        //Configuration 1
        if( (altitude>=1000) && (altitude<=10000) && (mach_no>=0.215) && (mach_no<=0.43) && (fabs(SimData->flightmodel2_controls_flap_handle_deploy_ratio-0.25)<=0.001) )
        {
            Idle_thrust= Compute_idle_conf1 (altitude, mach_no, row, column, table_conf1);
            RPWR=Idle_thrust*10;
```

Figure A.2: Idle thrust correction code part 1

```
if(PWR > RPWR + 1500.)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 4;
else if(PWR > RPWR + 500.)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 1;
else if(PWR > RPWR + 200.)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 0.25;
else if(PWR < RPWR + 100.)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 0.1;
else if(PWR > RPWR + 75.)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 0.08;
else if(PWR > RPWR + 50)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 0.06;
else if (PWR < RPWR - 20)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 0.03;
else if (PWR < RPWR)
    pA3MSData.qpac_airbus_dlrinterface_engoverride_idle_ovrd_value -= 0.01;
```

Figure A.3: Idle thrust correction code part 2

Finally, in figure A.4, it can be seen some other necessary functions of the A320FlightModel module, which are called in the main coded above. The first and the second one are used to obtain the idle thrust value from PEP tables for configuration 1 and clean configuration respectively, via an interpolation. The third one is used to obtain the correct overall drag coefficient from the tables, also with interpolation. Finally, the last one is used to compute the required fuselage drag coefficient to correct the overall drag coefficient of the aircraft.

```
double Compute_idle_conf1(int alt, double M, int row, int column, float table [13][11])
{
    double factor1=(table[row][column])+ ( (table[row][column+1]-table[row][column])* ( (alt-table[0][column])/(table[0][column+1]-table[0][column]) ) );
    double factor2=(table[row+1][column])+ ( (table[row+1][column+1]-table[row+1][column])* ( (alt-table[0][column])/(table[0][column+1]-table[0][column]) ) )

    double idle=(factor1)+ ( (factor2 - factor1)* ( (M-table[row][0])/(table[row+1][0]-table[row][0]) ) );

    return idle;
}

double Compute_idle_clean(int alt, double M, int row, int column, float table [33][41])
{
    double factor1=(table[row][column])+ ( (table[row][column+1]-table[row][column])* ( (alt-table[0][column])/(table[0][column+1]-table[0][column]) ) );
    double factor2=(table[row+1][column])+ ( (table[row+1][column+1]-table[row+1][column])* ( (alt-table[0][column])/(table[0][column+1]-table[0][column]) ) )

    double idle=(factor1)+ ( (factor2 - factor1)* ( (M-table[row][0])/(table[row+1][0]-table[row][0]) ) );

    return idle;
}


float Compute_cd(float cl, float M, int row, int column, float table [][31])
{
    double factor1=(table[row][column])+ ( (table[row][column+1]-table[row][column])* ( (M-table[0][column])/(table[0][column+1]-table[0][column]) ) );
    double factor2=(table[row+1][column])+ ( (table[row+1][column+1]-table[row+1][column])* ( (M-table[0][column])/(table[0][column+1]-table[0][column]) ) );

    float cd=(factor1)+ ( (factor2 - factor1)* ( (cl-table[row][0])/(table[row+1][0]-table[row][0]) ) );

    return cd;
}

double Compute_fus_cd(float s, float CD, float CDXP, float fus)
{
    double increment=-0.0000005*s+0.000255;
    CDXP=CDXP-((fus/0.001)*increment);
    double diff_cd=CD-CDXP;
    double value=(0.001*(diff_cd/increment));

    return value;
}
```

Figure A.4: *A320FlightModel* module functions